Ca' Foscari
University
of Venice

Laurea Magistrale
in
Scienze dell'economia
(Sviluppo economico e dell'impresa)
D.M. 270/2004

Tesi di Laurea

Atomic Arbitrage

Mercati finanziari decentralizzati e smart contracts per l'eliminazione dello slippage nell'arbitraggio tra cryptovalute

Relatore
Ch. Prof. Claudio Pizzi

Correlatore
Ch. Prof. Claudio Lucchese

Laureando
Matteo Pandolfi
Matricola 844227

Anno Accademico

2018/2019

a Satoshi

Sommario

ATOMIC ARBITRAGE

Matteo Pandolfi

Professor C. Pizzi Professor C. Lucchese

The purpose of this master thesis is to prove that using smart contracts in order to manage the cryptocurrencies' transactions that make up the arbitrage process, can remove the slippage risk component in the price of those financial instruments. The thesis, in addition to describing smart contracts and the Ethereum platform, illustrates the main features of a software, developed in the last two years, which is able to pull and analyze information fetched from decentralized financial markets and automate the arbitrage investment process. It will cover also the arbitrage algorithm and the programming framework used by the software and, as conclusion, the next development phases and possible implementations in order to optimize the arbitrage process. It will be also illustrated a statistical analysis of the software transactions among the last months in order to understand its profitability and the avoidance of the slippage risk in the arbitrage opportunities. The final achievement of this project is to allow investors to receive a periodic profit, proportionate to the invested capital, with a low-risk level.

Keywords: Blockchain, Smart Contracts, Atomic Arbitrage

Un ringraziamento particolare va al Professor Pizzi Claudio e al Professor Lucchese Claudio, per la pazienza e la disponibilità dimostrata.
Un ulteriore ringraziamento va a Francesco Bianchi, per la consulenza fornita in ambito LaTeXe formattazione di documenti accademici.
Ringrazio, per ultimo ma sicuramente non per importanza, William Bergamo, per aver creduto nell'idea dell'arbitraggio e per aver applicato la sua profonda conoscenza in ambito informatico e di blockchain per lo sviluppo di `At0m`.

Indice

Introduzione **1**

1 Blockchain e cryptovalute **5**

1.1 Decentralizzazione 5

1.1.1 Società cashless 5

1.1.2 Metodi di pagamento elettronico 6

1.2 Cryptovalute 8

1.3 I concetti generali della blockchain 11

1.3.1 Background storico 11

1.3.2 Definizione di blockchain 13

1.3.3 Ambiente: un network peer-to-peer distribuito 13

1.3.4 Analisi delle caratteristiche di una blockchain 14

1.3.5 Gestione delle transazioni 16

1.3.6 Processo di validazione 18

1.3.7 Hash dei dati . . . 19
1.3.8 Blocco . . . 19
1.3.9 Hash puzzle . . . 25
1.3.10 Mining . . . 26

2 Ethereum: Smart Contracts e piattaforma per DApps 29

2.1 Definizioni generali . . . 29
2.2 Obiettivi di Ethereum . . . 30
2.3 Caratteristiche principali dell'Ethereum protocol . . . 31
2.3.1 Semplicità . . . 31
2.3.2 Astrazione . . . 32
2.3.3 Modularità . . . 32
2.3.4 Flessibilità . . . 33
2.3.5 Assenza di discriminazione e censura . . . 33
2.4 Prime applicazioni client su Ethereum . . . 33
2.5 Accounts . . . 34
2.6 Transazioni . . . 36
2.7 Gas . . . 38
2.7.1 Unità di gas necessarie per le operazioni . . . 39
2.8 Valuta . . . 40
2.8.1 Prezzo del gas in Gwei . . . 41
2.9 EVM: Ethereum Virtual Machine . . . 42
2.9.1 Implementazioni . . . 44
2.9.2 Linguaggi di programmazione . . . 45
2.9.3 Debuggers . . . 46
2.10 Smart contracts . . . 46
2.10.1 Necessità di introduzione degli smart contracts . . . 46

2.10.2 Solidity: un linguaggio di programmazione per smart contracts . 47
2.11 Tokens 48
2.11.1 ERC-20 49
2.12 DApps: applicazioni decentralizzate 50
2.13 DExs: mercati decentralizzati 50
2.13.1 Mercati con *continuous-limit orderbook* 51
2.13.2 Tipologie di DEXs orderbook 51

3 Arbitraggio di prezzo tra valute **53**
3.1 Definizione di arbitraggio 53
3.2 Arbitraggio deterministico 54
3.3 Arbitraggio bilaterale 55
3.3.1 Arbitraggio bilaterale puro 55
3.3.2 Arbitraggio bilaterale con commissioni 57
3.3.3 Arbitraggio bilaterale in presenza di tassazione 64
3.3.4 Capital controls 65
3.3.5 Arbitraggio bilaterale in presenza del bid-ask spread 66
3.3.6 Arbitraggio bilaterale considerando tutte le condizioni 71
3.4 Arbitraggio triangolare 71
3.4.1 Arbitraggio triangolare in presenza di uno spread bid-ask 74
3.5 Arbitraggio multivalutario 76
3.5.1 Arbitraggio multivalutario in presenza dello spread bid-ask . . . 78
3.6 Rischi pratici dell'arbitraggio 80
3.6.1 Rischio di liquidità 80
3.6.2 Rischio di controparte 81
3.6.3 Rischio di esecuzione 82
3.6.4 Slippage 84

4 Project `At0m`: Atomic Arbitrage Bot **87**

4.1 Arbitraggio atomico . . . 88

4.2 Obiettivi di Project `At0m` . . . 89

4.3 Architettura del bot . . . 90

4.3.1 Rischi operativi di `At0m` . . . 93

4.3.2 Principali strumenti e librerie utilizzate . . . 94

4.4 Mercati decentralizzati (DExs) utilizzati . . . 96

4.4.1 Bancor Protocol . . . 96

4.4.2 Uniswap . . . 98

4.4.3 EtherDelta . . . 98

4.5 Risultati di Project `At0m` . . . 100

4.5.1 Esempio di una transazione di arbitraggio atomico . . . 102

5 Conclusioni **105**

A Concetti e meccanismi del Bitcoin **111**

A.1 Il ruolo della rete dei nodi (*node network*) . . . 111

A.1.1 Routers . . . 112

A.1.2 Wallets . . . 112

A.1.3 Miners . . . 112

A.1.4 Full nodes (nodi completi) . . . 113

A.2 Indirizzi (addresses) . . . 113

A.2.1 La costruzione dell'indirizzo . . . 114

A.2.2 Tipologie di indirizzi . . . 115

A.3 Transazioni . . . 115

A.4 Blockchain . . . 126

A.5 Consenso distribuito (*distributed consensus*) . . . 129

A.6 Mining . . . 131

A.7 Independent blockchain validation 136
A.8 Implicazioni per la sicurezza 136

B Piattaforma Ethereum e Smart Contracts **141**
B.1 Contratti 141
B.1.1 Contratto ricardiano 142
B.2 Smart contract 142
B.2.1 Esempi di contratto ricardiano 143
B.3 Ethereum 144
B.3.1 Concetti 145
B.3.2 Messaggi (messages) 147
B.3.3 Smart contracts 148

C Mercati decentralizzati **153**
C.1 Background 153
C.1.1 Exchanges centralizzati 154
C.1.2 Exchanges decentralizzati 156
C.1.3 Off-chain orderbook e On-chain orderbook 156
C.1.4 Vantaggi principali dei DEXs 157
C.1.5 Sfide principali dei DEXs 159
C.2 Bancor Protocol: Automated Liquidity Provider 163
C.2.1 Vantaggi degli Smart Tokens 164
C.2.2 Ecosistema di Bancor 166
C.3 UniSwap: un esempio 167

Bibliography **171**

Elenco delle figure

1.1 Diagramma transazione internazionale con carta 7
1.2 Autenticazione tramite firma digitale 18
1.3 Esempio di albero di Merkle 21
1.4 Informazioni contenute nel block header 21
1.5 Struttura a catena con i puntatori di ogni blocco collegati al blocco precedente 22
1.6 Block body: una struttura ad albero di Merkle 22
1.7 Risultato della modifica di una transazione su un blocco con struttura ad albero di Merkle 24

3.1 Effetto dell'arbitraggio bilaterale sulla domanda e sull'offerta 58
3.2 Linea di non-arbitraggio (arbitraggio bilaterale) 58
3.3 Effetto dell'arbitraggio bilaterale in presenza di commissioni 60
3.4 Linea di non-arbitraggio (arbitraggio bilaterale in presenza di commissioni) 61

3.5 Linea di non-arbitraggio (arbitraggio bilaterale in presenza di commissioni progressive) . . . 63
3.6 Linea di non-arbitraggio (arbitraggio bilaterale in presenza di controllo sui movimenti dei capitali) . . . 66
3.7 Effetto dell'arbitraggio bilaterale in presenza di uno spread *bid-ask* . . . 68
3.8 Condizione di non-arbitraggio in presenza di un bid-ask spread . . . 69
3.9 Effetto dell'arbitraggio triangolare sulla domanda e sull'offerta . . . 75

4.1 Architettura di `At0m` . . . 90
4.2 Linea di non-arbitraggio atomico . . . 93
4.3 Riepilogo dell'andamento del wallet di `At0m` . . . 100
4.4 Panoramica sulle operazioni di arbitraggio avvenute con successo . . . 101
4.5 Informazioni generali sull'operazione di arbitraggio atomico . . . 103
4.6 Dettaglio dei tokens scambiati nell'operazione di arbitraggio atomico . . . 103
4.7 Transazioni interne dell'operazione di arbitraggio atomico . . . 104

5.1 Rappresentazione delle fasi di sviluppo di `At0m` . . . 107

A.1 Spending e mixing del valore (in bitcoin) . . . 118
A.2 Struttura di collegamento dei blocchi . . . 129

C.1 Struttura degli exchange centralizzati . . . 155
C.2 Perdite derivanti da hacks negli exchange centralizzati . . . 155
C.3 Maggiori 5 DEXs per volumi giornalieri . . . 157
C.4 Struttura degli exchange decentralizzati . . . 158
C.5 Schema grafico del esempio descritto nella Sezione C.3 . . . 169

Elenco delle tabelle

2.1 Operazioni più significative della EVM 40
2.2 Sottodenominazioni di Ether (prezzo rilevato 20/06/2019 41
2.3 Fasce di prezzo per le transazioni 41

4.1 Riepilogo dell'andamento dei profitti di At0m (1 ETH = $268.12, 20/06/2019) 100

A.1 Struttura delle transazioni [46] 117
A.2 Inputs e Outputs 118
A.3 Input 119
A.4 Esecuzione dello script 120
A.5 Struttura del blocco 127
A.6 Struttura della testata del blocco 127

B.1 Struttura di un account con proprietà esterna 146
B.2 Struttura di una transazione 147
B.3 Struttura di un messaggio [81] 148

B.4 Istruzioni principali della EVM e il loro relativo costo 150

Introduzione

L'avvento, negli ultimi anni, della blockchain e delle cryptovalute ha avuto un forte impatto anche, e soprattutto, nel mondo dei mercati finanziari. La frenesia iniziale generata da questo nuovo mercato, rappresentata da un vertiginoso incremento generalizzato del valore di mercato delle cryptovalute, ha fatto rivolgere l'attenzione di molti esperti provenienti da ambiti più disparati, ma ha creato anche molta confusione riguardo questa neonata tecnologia, in particolare per quanto riguarda la sua applicazione pratica. La tecnologia blockchain permette, nella sua descrizione più basilare, di costruire registri digitali pubblici e inalterabili, che come conseguenza possono portare alla disintermediazione di molti attori di settori come assicurazioni, banche e logistica. La decentralizzazione fornita dalla blockchain, inoltre, permette di eliminare il rischio di avere un singolo punto critico di vulnerabilità dei dati; in un mondo come quello attuale, dove l'alto livello di dipendenza dalla tecnologia e da Internet ha portato a nuovi modelli di business per le organizzazioni, questa caratteristica della blockchain potrebbe permettere di eliminare una serie di rischi legati al controllo dei dati, in particolare

quelli sensibili, che, giorno dopo giorno, continuano ad aumentare esponenzialmente.
La nascita della prima cryptovaluta, il *Bitcoin*, avviene in concomitanza con la crisi finanziaria del 2008, quasi in risposta agli errori e alle speculazioni effettuate dai colossi della finanza; la proposta di una moneta digitale supportata da un sistema di registrazione delle transazioni che permettesse, grazie ad una struttura di dati la cui certezza e inalterabilità è assicurata da una serie di regole crittografiche, lo scambio di valore senza la necessità di affidarsi ad un intermediario non ha destato immediatamente l'attenzione del grande pubblico, finendo con l'essere catalogata come denaro prettamente utilizzato per traffici illegali nel *Deep Web*. Nonostante ciò, il gruppo di ricercatori e sviluppatori interessati alla blockchain e ad una sua applicazione legale nel mondo reale, hanno continuato il suo sviluppo e l'avvento di nuovi ecosistemi come *Ethereum* hanno permesso di introdurre concetti e logiche del tutto nuovi, anche se basati su ricerche e proposte effettuate ben prima dell'avvento e dell'affermazione di Internet. Fra queste possiamo individuare il concetto di *Smart Contract*, teorizzati ed oggetto di sperimentazioni negli anni 90 [1], il quale ha introdotto una serie di applicazioni pratiche che, in particolare per la finanza e per il progetto che verrà presentato in questo trattato, hanno permesso di dimostrare la tesi che verrà presentata. Uno smart contract è la "traduzione" o "trasposizione" in codice di un contratto, il quale permette di verificare in modo automatico l'avverarsi di specifiche condizioni e conseguentemente eseguire in automatico una o più azioni, o dare disposizione affinché vengano eseguite determinate azioni nel momento in cui le condizioni desiderate tra le parti vengono raggiunte e verificate.
Le fondamenta del progetto che verrà presentato in questa tesi vengono poste all'incirca tre anni fa, quando, grazie alla collaborazione di due studenti universitari, rispettivamente di informatica e di economia, vengono notate le discrepanze di prezzo delle cryptovalute. Viene deciso così che l'arbitraggio è la nicchia su cui si vuole puntare. Il termine arbitraggio si riferisce a un processo mediante il quale un investitore sfrutta

una mancata corrispondenza tra i prezzi di due o più asset finanziari, in questo caso cryptovalute; comprando e vendendo queste valute, l'investitore può ottenere un profitto privo di rischio (come verrà descritto nel capitolo 3). Tale mancata corrispondenza di solito permane sul mercato solo per un tempo limitato in quanto, in particolare nei mercati finanziari tradizionali, c'è un elevato numero di operatori attivi nella ricerca di tali opportunità, quindi solo traders con attrezzature avanzate sono solitamente in grado di trarre vantaggio di tali disalineamenti.

Nei successivi due anni viene così attuato lo sviluppo di un software che permette di interagire con i mercati delle cryptovalute e con la blockchain: viene ideato inizialmente un sistema di arbitraggio triangolare[1] e, in parallelo, viene effettuata, con il supporto di un altro programmatore, un'attività di investimento tramite trading algoritmico, sfruttando la struttura già creata per interfacciarsi ai mercati delle cryptovalute, permettendo di testare le teorie formulate e acquisire altre conoscenze sui meccanismi e sulle dinamiche di questo mercato. Ciò ha poi portato all'intuizione che sta alla base di questa tesi: osservando una delle possibilità data dagli smart contract, ossia quella di poter includere multiple operazioni all'interno di un'unica transazione, è stato ipotizzato che tale caratteristica poteva avere le potenzialità di risolvere una delle componenti di rischio legate alla pratica dell'arbitraggio, il cosiddetto *slippage*, cioè il rischio dato dalla probabilità che il prezzo a cui un'operazione viene eseguita sia diverso dal prezzo atteso dall'investitore, provocando così una perdita. Grazie alle conoscenze in campo informatico e, in particolare, nella creazione degli smart contract, è stato possibile applicare nel mondo reale tale teoria e dimostrare, come verrà spiegato in questa tesi, l'effettiva profittabilità di questa nuova strategia di investimento.

Verranno quindi spiegati, in questa tesi, i tratti principali della blockchain, sia dal punto di vista teorico che dal lato tecnico, per poter poi introdurre anche *Ethereum*, la piattaforma che permette l'esecuzione degli smart contracts e di conseguenza anche l'arbitraggio

[1] Un approfondimento legato all'arbitraggio triangolare verrà trattato nel Capitolo 3 di questa tesi.

(come verrà illustrato nel Capitolo 2); Infine, grazie ai concetti spiegati nei capitoli precedenti, sarà possibile introdurre il progetto, denominato "`At0m`", il quale consiste in un sistema automatizzato di investimento che permette di effettuare arbitraggio atomico[2] (Capitolo 4). Questo permette quindi, come già accennato, di eliminare la componente di rischio apportata dallo slippage ed di introdurre una serie di logiche prima impossibili da attuare nel mondo della finanza tradizionale.

In questo modo si è potuto dimostrare che, legando una nuova tecnologia a concetti di finanza pre-esistenti, nonostante i miglioramenti effettuabili sia molteplici, sia per quanto riguarda At0m che per l'ecosistema della blockchain in generale, è possibile affinare ed ottimizzare alcune strategie di investimento, introducendo nuove logiche che attualmente sono solo agli albori della loro massima espressione.

[2] Definito atomico in quanto le due operazione che compongono l'arbitraggio non sono divisibili, caratteristica, presa in prestito dalla fisica, in comune con quella di un atomo.

Capitolo 1

Blockchain e cryptovalute

1.1 Decentralizzazione

1.1.1 Società cashless

Nell'anno 2017, il 53% della popolazione europea fra i 16 e i 74 anni ha comprato beni o servizi utilizzando Internet. Dieci anni prima, nel 2007, la percentuale era del 30%. Internet permette ai suoi utilizzatori di inviare istantaneamente pagamenti dal proprio pc o smartphone e i venditori possono ricevere tale denaro nel giro di pochi giorni od ore pur risiedendo in paesi lontani. Ma i pagamenti cashless sul Web non sono l'unica cosa ad essere popolare. Nell'agosto 2016, l'80% delle transazioni in Svezia è stato fatto tramite pagamento elettronico [2]. La crescente popolarità dei pagamenti tramite Internet o carta ha portato alla forte riduzione dell'utilizzo di denaro contante e questo è stato ulteriormente confermato da un altro trend. Le leggi approvate in Italia (2011), in Spagna (2012) e in Francia (2015) proibiscono l'uso di contanti per transazioni che superano un determinato limite [3].
I sopracitati limiti sono rispettivamente 3000€, 2500€, 1000€. Nel caso dell'Italia e della Spagna, tali leggi sono mirate a ridurre l'evasione fiscale, mentre per quanto riguarda

la Francia, l'obiettivo è quello di bloccare il riciclaggio del denaro e il finanziamento al terrorismo. Nel caso in cui si verifichi un periodo di forte e non controllata inflazione, o se i governi imporranno ulteriori restrizioni, le transazioni tramite denaro contante possono diventare molto sconvenienti e poco pratiche. Il risultato della combinazione di tali fattori ha portato a sostituire il tradizionale trasferimento fisico del denaro fra due parti, un mittente ed un ricevente, con un trasferimento digitale di dati fra tre parti, un mandante, un mediatore e un ricevente.

1.1.2 Metodi di pagamento elettronico

Le transazioni finanziarie elettroniche che permettono ad una parte di inviare somme di denaro ad un'altra sono processate, validate e confermate da uno o più mediatori. Un esempio di transazione con un solo mediatore è il trasferimento di denaro da un conto corrente A ad un altro conto B all'interno della stessa banca. Quest'ultima, verifica se nel conto A è presente una somma sufficiente a effettuare la transazione e si occupa di coordinare il processo di contabilizzazione dell'operazione. Un altro esempio può coinvolgere invece due mediatori: una persona effettua un pagamento, tramite il conto corrente appartenente ad una banca, ad un altra persona il cui conto corrente appartiene ad una differente banca; entrambe le banche partecipano al coordinamento dell'operazione di pagamento. Nella pratica però, le comunicazioni infra-bancarie coinvolgono ulteriori mediatori: la rete SWIFT, e i circuiti Visa o MasterCard. Nei pagamenti internazionali, addirittura più di tre mediatori possono partecipare al trasferimento elettronico di denaro, come viene evidenziato in Figura 1.1.

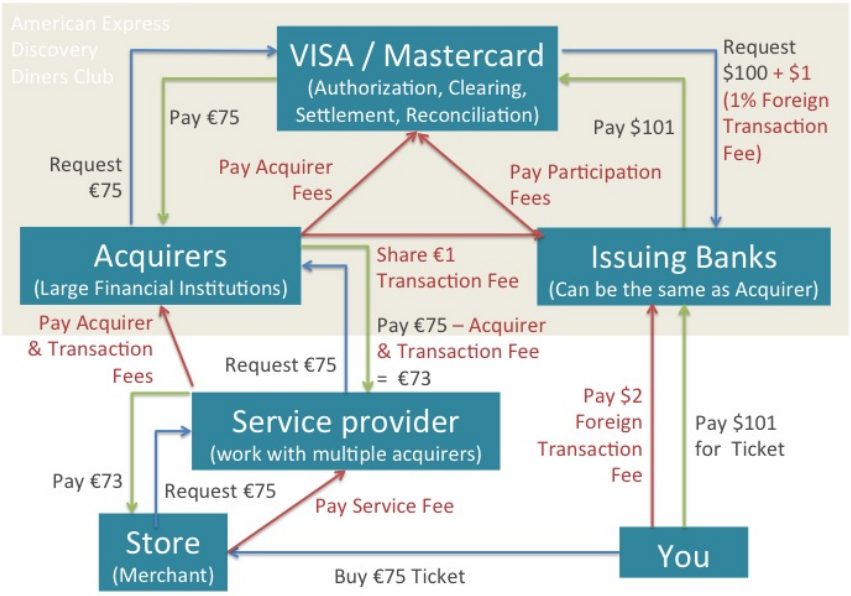

Figura 1.1: Diagramma transazione internazionale con carta [4]

La figura 1.1 mostra l'acquisto fittizio di un biglietto per 75€. Prima che ogni transazione venga effettuata, la banca emittente della carta utilizzata e gli intermediari (spesso istituti finanziari) devono pagare un canone per la partecipazione al circuito VISA o MasterCard. Anche il venditore del biglietto deve pagare una fee ad un fornitore di servizi di pagamento, il quale si appoggia agli istituti finanziari, per avere accesso al network VISA o MasterCard in modo tale da poter ricevere il denaro. Per questo esempio utilizzeremo VISA come fornitore del circuito di pagamento. Nel caso in cui la banca offra una carta senza commissioni internazionali, viene caricata un'ulteriore fee al commerciante, che si ritrova a pagare sia la commissione per il circuito VISA che una commissione per ogni transazione per accettare il pagamento. Quest'ultima include una parte fissa per gli intermediari finanziari e una parte che viene divisa fra la banca emittente e l'istituto finanziario.

Seguendo l'esempio riportato in Figura 1.1, il commerciante europeo riceverebbe 73€

a fronte di un pagamento di 75€[1]. I mediatori finanziari hanno quindi un incentivo economico: questo proviene dalle commissioni pagate dagli utilizzatori del servizio e dai commercianti. C'è il rischio che, nel caso in cui non si presentino sufficienti concorrenti sul mercato dei mediatori, si possa ricadere in un monopolio o una forma di cartello. Questi sistemi di pagamento presuppongono la fiducia nelle autorità che li autorizzano; l'utilizzatore deve fidarsi che ogni componente della catena di mediatori autorizzerà la transazione e trasferirà il denaro, pagandone il servizio. In modo simile, il commerciante dovrà affidarsi a questa catena dei mediatori per ricevere l'ammontare concordato. Tuttavia, i sistemi digitali di pagamento basati su questa fiducia non sono completamente affidabili. Decisioni politiche, o anche strategie di business, possono creare pressioni sugli intermediari che offrono questo servizio e spingerli a cambiare il sistema in modo da sfavorire alcuni utenti. Nel 2010, dopo l'uscita del *Cablegate* di Wikileaks, compagnie come Bank of America, VISA, MasterCard, PayPal e Western Union imposero un blocco sui sistemi digitali di pagamento di Wikileaks, sopprimendone così il 95% del suo profitto. Sia la monopolizzazione che la componente di fiducia nell'autorità sono fortemente presenti in molte altre società digitali non-finanziarie come motori di ricerca, piattaforme per l'e-commerce e social networks.

1.2 Cryptovalute

Il concetto di cryptovaluta è stato introdotto nel mercato nella seconda metà del XXI secolo, in risposta ad un sistema monetario e bancario altamente centralizzato; ciò può essere anche visto come una reazione alla crisi finanziaria globale del 2008, la quale ha portato ad un senso generale di sfiducia nei confronti del governo degli Stati Uniti e delle istituzioni finanziare e conseguentemente ha permesso l'avvio di tale rivoluzione

[1]Normalmente i commercianti assorbono semplicemente la commissione; ma in molti paesi è consueto far pagare un surplus se si vuole pagare con carta di credito, per i motivi descritti sopra. Un altro modo per scoraggiare il pagamento con carte di credito è quello di offrire uno sconto nel caso di pagamento in contanti.

[5]. Al giorno d'oggi esistono più di un migliaio di varie forme di cryptovalute [6], che rappresentano una sfida ai fondamentali del sistema monetario con valute *fiat*[2] in quanto forniscono un sistema di pagamento completamente indipendente da enti governativi o dalle cosiddette *clearing houses*, cioè le camere di compensazione[3]. Le cryptovalute sono basate su una tecnologia chiamata *blockchain*, la quale permette di effettuare transazioni la cui validità viene assicurata da una serie di leggi crittografiche piuttosto che da una terza parte in cui bisogna riporre la propria fiducia [7]. Dal momento che le cryptovalute non si basano, attualmente, su un tasso di interesse, i modelli di valutazione basati sui tassi di interesse diventano privi di significato [8]; in tale situazione, il prezzo diventa dipendente dalla valutazione che gli utenti conferiscono alle cryptovalute, in base alle informazioni a loro disponibili [8]. Tenendo presente tale concetto, è possibile considerare il prezzo delle cryptovalute basato su una valutazione della domanda e dell'offerta, e in quanto il numero di *coins* è limitato, ad un incremento della domanda corrisponderà conseguentemente da un incremento del prezzo. Rimane ancora aperta la questione se le cryptovalute come Bitcoin o Ethereum debbano essere considerate valute o strumenti di investimento. Come viene esposto in Glaser et al. [8], la maggior parte dei nuovi utenti di Bitcoin ritengono la cryptovaluta più un investimento che un metodo di pagamento. La questione su come classificare le cryptovalute è stata trattata anche da Dyhrberg [9], la quale ha analizzato in che misura il prezzo del Bitcoin sia correlato con il prezzo dell'oro.

In accordo con la Banca Centrale Europea [10], la tassonomia delle valute virtuali può essere divisa in tre categorie principali. La prima è *closed virtual currency schemes*, ed un esempio può essere quello del *World of Warcraft* (WoW) *Gold*, il quale può essere comprato, venduto, guadagnato ed utilizzato solamente all'interno del mondo virtuale.

[2] Con valute *fiat* si intendono tutte quelle valute che vengono emesse da un ente governativo ma che non hanno alcun controvalore con una *commodity* fisica, come l'oro o l'argento.

[3] La compensazione è un meccanismo che permette alle banche e alle istituzioni finanziarie di regolare tra loro i rapporti di dare e avere generati da transazioni finanziarie effettuate sui mercati o di scambio di assegni o denaro tra banche.

La seconda tipologia è *virtual currency schemes with unidirectional flow*, nelle quali la moneta virtuale può essere utilizzata all'interno del mondo virtuale (come delle *coins* di un gioco per smartphone) ma viene acquistata con valute tradizionali; tuttavia, non può essere riconvertita in valuta *fiat*. Le valute virtuali che verranno trattate in questa tesi fanno parte della terza tipologia, cioè *virtual currency schemes with bidirectional flow*. Gli utenti hanno la possibilità di convertire il loro denaro in entrambe le direzioni, al tasso di conversione attuale e possono essere utilizzate per comprare beni e servizi sia nel mondo virtuale che in quello reale [10]. Un sotto insieme delle *virtual currency schemes with bidirectional flow* sono le cryptovalute, intese come metodo di scambio in un contesto digitale. Alla base delle crytpovalute c'è la blockchain, una tecnologia che utilizza (come vedremo nei paragrafi successivi) un *decentralized ledger* come strumento di verifica delle transazioni. Il fatto che essa utilizzi la crittografia per verificare e validare i dati al suo interno permette di non avere più la necessità di affidarsi ad una terza parte. La blockchain è stata identificata come tecnologia che permetterà di cambiare il mondo delle transazioni finanziarie in termini di sicurezza, efficienza, costi, intermediazione, privacy, trasparenza ed immutabilità. Grazie alle sue caratteristiche, la blockchain può permettere di creare un totalmente nuovo modello finanziario, soprattutto se lo si compara a quello tradizionale; come conseguenza, vari settori professionali potrebbero essere trasformati, in particolare per aspetti come la sicurezza, i metodi di pagamento e i costi di commissione. Una delle caratteristiche più importanti introdotte da questa nuova tecnologia riguarda la relazione fra gli utenti della blockchain e gli intermediari tradizionali, come notai, banche e i gestori dei diritti d'autore; detto ciò, in un potenziale scenario dove la blockchain viene adottata a livello mondiale, il ruolo di intermediario potrebbe essere completamente ridefinito o addirittura eliminato. Grazie a questa caratteristica di disintermediazione, la blockchain è spesso definita come una tecnologia *diruptive*. Un'altra importante caratteristica riguarda il mondo della finanza; infatti, sia il mercato valutario, quello azionario, quello obbligazionario e i derivati possono essere

gestiti e mantenuti attraverso la blockchain. Di conseguenza, oltre che ai vantaggi appena citati, sarà presente anche la caratteristica della dematerializzazione, che con questa tecnologia potrebbe portare ad una completa eliminazione della carta stampata in favore di un ecosistema completamente digitale. Come risultato, i costi risparmiati potrebbero essere rilevanti sia per il *provider* che per gli utenti dei servizi basati sulla blockchain. Anche se la blockchain promette di portare una maggiore efficienza o una riduzione dei costi, essa ha alcuni rischi intrinseci, come verrà analizzato nel corso di questa tesi. È necessario che le aziende comprendano questi rischi e le opportune misure di salvaguardia per trarre vantaggio da questa tecnologia. Inoltre, è importante comprendere l'evoluzione della normativa a riguardo e le sue implicazioni.

1.3 I concetti generali della blockchain

Nelle seguenti sezioni verranno descritti i concetti generali di una blockchain, in modo tale da illustrare il background storico, le definizioni più significative e i razionali tecnici dei vantaggi appena descritti. E' importante definire il background storico della blockchain per poter comprendere completamente i miglioramenti effettuati durante gli ultimi anni.

1.3.1 Background storico

La prima applicazione di una blockchain è avvenuta tramite Bitcoin; è quindi difficile discutere la storia di questa tecnologia senza dedicare una parte della discussione a Bitcoin.

Gli inizi con Bitcoin

La tecnologia blockchain fa il suo debutto pubblico nel 2008, quando un gruppo, o una persona singola, in modo anonimo e attraverso l'uso dello pseudonimo di *Satoshi*

Nakamoto, rilascia il white paper *"Bitcoin: A Peer to Peer Electronic Cash System* [7]. Successivamente al rilascio del paper, Bitcoin viene offerto, nel 2009, ad una comunità open source di ricercatori.

La blockchain separata da Bitcoin

Anche al giorno d'oggi, c'è chi ritiene che Bitcoin e blockchain siano la stessa cosa, anche se concretamente non è così. In contrasto a ciò, coloro che, nel 2014, realizzarono che la blockchain poteva avere applicazioni diverse dalle cyptovalute, iniziarono ad investire su tale tecnologia. Come risultato, diverse nuove blockchain iniziarono ad apparire sul mercato, con lo scopo di rivoluzionare supply chain, la sanità, il settore assicurativo, i trasporti, i sistemi di votazione, la gestione dei contratti e altro ancora. In aggiunta a ciò, anche il settore finanziario ha iniziato un piano di investimento nei confronti di questa tecnologia; al momento, circa il 15% delle istituzioni finanziarie utilizzano la blockchain [11].

L'arrivo di Ethereum e gli smart contracts

Vitalik Buterin, il co-fondatore di Ethereum e Bitcoin magazine, fu inoltre uno dei collaboratori iniziali a contribuire allo sviluppo della codebase[4] di Bitcoin, ma ne rimase deluso dalle limitazioni nella programmabilità. Riscontrando una certa resistenza della community di Bitcoin, Buterin decise di sviluppare una seconda blockchain pubblica chiamata Ethereum, lanciata nel 2015.

La principale differenza sta nel fatto che Ethereum può registrare altri assets come prestiti o contratti, e non solo delle transazioni finanziarie. Ethereum può essere utilizzato per creare degli "smart contracts", i quali permettono di eseguire automaticamente

[4]Il termine codebase è usato in programmazione per indicare l'intera collezione di codice sorgente usata per costruire una particolare applicazione o un particolare componente.

alcune funzioni in base a dei criteri stabiliti nel contratto e che vengono registrati nella blockchain di Ethereum. Tutto ciò verrà approfondito nel Capitolo 2.

1.3.2 Definizione di blockchain

La grande quantità di documentazione pubblicata negli ultimi anni riguardo la blockchain ha creato uno stato di confusione generale riguardo la sua definizione; tuttavia, la maggior parte delle risorse disponibili trovano un accordo con tale definizione:

La *blockchain* (letteralmente, catena di blocchi) può essere paragonata ad un libro mastro, condiviso con tutta la rete e crittograficamente immutabile, dove le transazioni vengono permanentemente registrate in blocchi sequenziali.

É importante sottolineare un aspetto che molto spesso porta ad una considerazione sbagliata di questa tecnologia: la blockchain non è un database, nel senso tradizionale, poiché non è possibile archiviare qualsiasi cosa al suo interno. L'archiviazione di immagini o file PDF, ad esempio, non è ammessa. Il *ledger* è distribuito e tutti all'interno del network della blockchain ne hanno una copia; di conseguenza, la blockchain deve mantenere delle dimensioni tali da poter permettere un certo grado di scalabilità[5].

1.3.3 Ambiente: un network peer-to-peer distribuito

Una blockchain risiede in un particolare ambiente distribuito, una rete peer-to-peer. Nel dettaglio, questa rete è composta da macchine individuali, chiamate nodi, le quali forniscono potenza computazionale ai restanti membri del network, senza la presenza di un'autorità centrale. Figurativamente, è simile ad un ecosistema completamente

[5] Nell'ingegneria del software, nelle telecomunicazioni, in informatica e in altre discipline, la scalabilità denota in genere la capacità di un sistema di aumentare o diminuire di scala in funzione delle necessità e disponibilità. Un sistema che gode di questa proprietà viene detto scalabile. La scalabilità può essere di carico, (cioè quando un sistema può aumentare le proprie prestazioni in funzione della potenza di calcolo complessiva che viene dedicata alla sua esecuzione) geografica (cioè quando un sistema mantiene inalterata la sua usabilità e utilità indipendentemente dalla distanza fisica dei suoi utenti o delle sue risorse) e amministrativa (quando un sistema mantiene inalterata la sua gestibilità indipendentemente da quante organizzazioni lo utilizzano.

democratizzato, dove ogni nodo è uguale a tutti gli altri sia in termini di diritti che di ruolo all'interno della rete; come risultato, potenzialmente, ogni nodo può scegliere autonomamente se consumare o fornire le risorse, senza alcuna limitazione.

I sistemi peer-to-peer devono soddisfare alcuni requisiti per poter operare correttamente:

- La coordinazione deve avvenire fra i membri della rete;
- Dev'essere presente un protocollo di comunicazione per inviare, ricevere e processare messaggi;
- La rete deve avere un adeguata capacità computazionale per poter risolvere un problema di calcolo;
- Dev'esserci la garanzia che il sistema riesca a rispondere adeguatamente a diversi problemi di sicurezza.

1.3.4 Analisi delle caratteristiche di una blockchain

Questa tecnologia è sia un'innovazione economica, una nuova architettura per permettere lo scambio di informazioni fra umani e macchine senza il vincolo di doversi fidare della controparte, sia un'innovazione tecnologica, un moderno *ledger* decentralizzato che assicura la fiducia tra le parti [12] in quanto tale tecnologia ha la possibilità di offrire una piattaforma di registrazione delle transazioni e la condivisione dei dati fra i partecipanti in modo efficiente, trasparente e completamente verificabile [13].

Ledger

Una blockchain è quindi un registro dei conti digitale, analogo al corrispettivo fisico, dove tutte le transazioni vengono registrate. In altre parole, lo scopo della blockchain

è quello di mantenere le informazioni storiche che riguardano tutte le transazioni effettuate, dal *genesis block* (cioè il primo blocco generato in una blockchain) e l'ultimo blocco generato; da qui il nome blockchain. I dati vengono registrati in ordine temporale, come su un libro contabile o un file Excel ordinato cronologicamente.

Distribuito e condiviso

Distribuito e condiviso sono due proprietà utilizzate per descrivere il *ledger*: la prima caratteristica definisce il fatto che le informazioni vengono effettivamente distribuite in ogni nodo; la seconda invece denota il fatto che, teoricamente, le informazioni sono identiche per ogni nodo della rete, in ogni instante.

Crittograficamente immutabile

La proprietà di essere crittograficamente immutabile si riferisce al fatto che, ogni volta che un nuovo blocco di transazioni viene aggiunto alla catena, viene reso immutabile, garantendone l'integrità dei dati e rendendo le informazioni contenute nel *ledger* completamente affidabili; il tutto tramite un algoritmo crittografico.

L'integrità e l'affidabilità sono due facce della stessa moneta. Da un lato, l'integrità del sistema significa assumere che questo operi senza avere la possibilità di essere compromesso, libero da ogni manipolazione non autorizzata. Dall'altro lato, l'affidabilità del sistema viene fornita in anticipo e cambia a seconda dei risultati delle interazioni con esso. In conclusione, in una rete peer-to-peer, non vi può essere fiducia fra gli utenti senza preservare l'integrità globale del sistema.

L'acquisizione e il mantenimento dell'integrità dei dati dipende dai seguenti fattori [14]:

- trasparenza sul numero dei nodi;
- trasparenza sull'affidabilità dei nodi;

- conoscenza del numero di fallimenti tecnici, componenti che non funzionano a dovere o che producono risultati errati;
- trasparenza sui nodi "disonesti", i quali attaccano intenzionalmente il sistema, abbassandone il grado di affidabilità.

Il problema che viene risolto in una blockchain è quello di acquisire e mantenere l'integrità dei dati in un sistema peer-to-peer composto da un numero sconosciuto di nodi con integrità indefinita. Questa classe di problemi è stata largamente discussa dalla comunità tecnologica online, ed è conosciuta con il nome *Byzantine General problem*[6].

1.3.5 Gestione delle transazioni

Prima di aggiungere un blocco di transazioni alla catena, dev'essere fornita la proprietà del bene o del servizio. La prova della proprietà richiede tre parametri:

- l'identificazione del proprietario;
- l'identificazione del bene/servizio posseduto;
- la relazione fra il proprietario e il bene/servizio posseduto.

Quest'ultima è ottenuta grazie al *ledger*, dove viene registrata l'associazione fra il proprietario e il bene posseduto; tale relazione è quindi intrinseca alla blockchain. Ciò richiede l'implementazione di due proprietà: *autenticazione* e *autorizzazione*.

[6]Il *Byzantine General problem* è un problema informatico su come raggiungere consenso in situazioni in cui è possibile la presenza di errori. Il problema consiste nel trovare un accordo, comunicando solo tramite messaggi, tra componenti diversi nel caso in cui siano presenti informazioni discordanti. Informalmente il problema è esemplificato dalla situazione in cui tre o più generali bizantini debbano decidere se attaccare o ritirarsi dato un ordine da un comandante superiore. Uno o più dei generali potrebbero essere dei traditori con l'intenzione di confondere gli altri, quindi potrebbe verificarsi il caso in cui il comandante dia ordini discordanti ai generali oppure il caso in cui uno dei generali comunichi ai propri colleghi un ordine differente da quello impartito dal comandante. La soluzione al problema permette ai generali leali di evitare queste trappole.

Autenticazione

Nel contesto dei sistemi informatici, l'autenticazione è il processo che assicura e conferma l'identità di un utente. L'autenticazione è la garanzia che un utente è chi dichiara di essere.

Come prima cosa l'utente deve fornire i propri diritti di accesso e l'identità: effettuando il login in un computer, gli utenti sono soliti inserire il proprio username e password. Questa combinazione, che dev'essere assegnata ad ogni utente, permette di autenticarne l'accesso. Tuttavia, questo tipo di autenticazione può essere bypassata da un agente malevolo, come un hacker.

Smarrire la password in un sistema come quello della blockchain comporta la potenziale perdita dei fondi contenuti nel proprio *wallet* digitale, dove sono contenuti tutti i fondi dell'utente; e una volta che i fondi vengono perduti, l'operazione non è più reversibile.

Autorizzazione

L'autorizzazione è un meccanismo di sicurezza utilizzato per determinare i privilegi degli utenti o il loro livello di accesso a determinate risorse del sistema, inclusi file, servizi, dati e applicazioni. L'autorizzazione è normalmente preceduta dall'autenticazione per verificare l'identità dell'utente. Durante l'autorizzazione, un sistema verifica i permessi di un utente già identificato, garantendo o rifiutando l'accesso a determinate risorse.

Effettuare una transazione finanziaria è un operazione che richiede un elevato livello di sicurezza; per questo dev'essere stabilita una politica di accesso. L'autorizzazione nei sistemi basati sulla blockchain viene acquisita grazie all'utilizzo di una firma digitale (Figura 1.2).

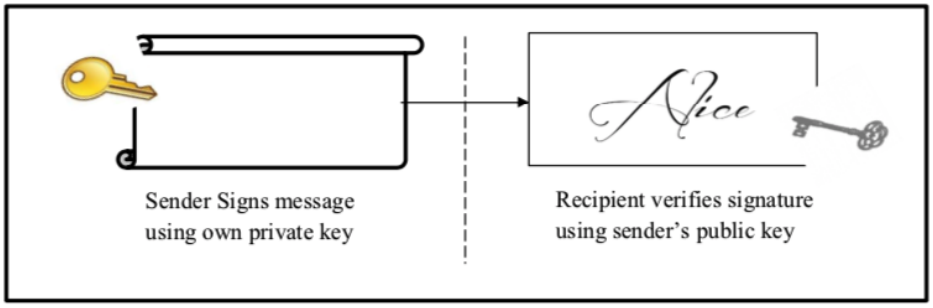

Figura 1.2: Autenticazione tramite firma digitale

É possibile approfondire i meccanismi di autorizzazione delle transazioni su blockchain nell'Appendice A.

1.3.6 Processo di validazione

I nodi sono delle entità appartenenti ad una blockchain che ne rappresentano gli utenti. I nodi agiscono come dei testimoni, dichiarando se una transazione è stata realmente effettuata dal mittente x al ricevente y con un importo z. Tale testimonianza garantisce l'assenza di frodi all'interno del network della blockchain. Come risultato, maggiore è il numero di nodi e maggiore sarà l'affidabilità di tale processo.

Il processo di validazione della transazione, che coinvolge tutti i nodi, è possibile grazie al protocollo fornito dalla blockchain. Tale protocollo ha il compito di includere tutti i nodi nella decisione, ad esempio se considerare una transazione valida oppure no. Il protocollo permette di coordinare il processo di controllo del proprio *ledger* per ogni nodo nella rete; facendo ciò, se la maggior parte dei nodi è in accordo sullo stato della transazione, quest'ultima viene memorizzata permanentemente nella blockchain. Lo stato di una transazione può essere *successful*, quando viene approvata, oppure *rejected*, quando viene considerata errata o incongruente.

Per esempio, in una transazione finanziaria, un nodo deve verificare chi sia il soggetto che sta inviando il denaro, dopo che esso ha effettuato il processo di autenticazione

ed autorizzazione e che esso abbia a disposizione un importo di denaro maggiore della quantità che sta cercando di inviare.

1.3.7 Hash dei dati

Un sistema basato sulla blockchain dipende fortemente dalle sue funzioni crittografiche di hash[7]. Tali funzioni, se scelte propriamente, sono:

- applicabili ad ogni tipo di dati;
- deterministiche;
- pseudo-random;
- unidirezionali;
- resistenti alle collisioni[8].

L'obiettivo di queste funzioni è di identificare la transazione, comparandone il risultato della funzione di hash, senza dover comparane direttamente i dati. Infatti, comparare direttamente le informazioni della transazione è un processo lungo e computazionalmente pesante. Il processo di verifica della validità dei dati è un elemento fondamentale; se i dati sono assicurati al 100%, gli utenti sono protetti da ogni tipo di frode o attacco informatico.

1.3.8 Blocco

Un blocco è un gruppo di transazioni ordinate cronologicamente, o perlomeno nell'ordine cronologico in cui i nodi miners, come vedremo nel paragrafo 1.3.10, hanno

[7] Una funzione crittografica di hash è una funzione che mappa un messaggio arbitrariamente lungo in una stringa di lunghezza prefissata, cercando di far in modo che da questa stringa non si possa risalire al messaggio che l'ha generata (a meno che non si sia in possesso della chiave con cui è stata generata). La lunghezza della stringa finale è direttamente correlata con la sicurezza della funzione di hash, perché più una stringa è lunga, minore sarà la probabilità di trovare due messaggi con lo stesso digest, con la stessa stringa finale.

[8] In crittografia, una collisione hash è una situazione che avviene quando due diversi input producono lo stesso output tramite una funzione hash.

raggiunto un accordo sull'organizzazione delle transazioni. Ogni blocco include, fra le varie informazioni, l'hash del blocco precedente. Ogni blocco è composto da un "*block header*" e un "*block body*". Ogni blockchain ha una propria struttura per il blocco, ma le caratterstiche fondamentali sono sempre le stesse.

Merkle tree

Prima di analizzare la struttura di un blocco, è opportuno introdurre brevemente il concetto di Merkle tree. Un albero di Merkle, o albero di hash, è un albero per lo più binario, le cui ramificazioni sono due a ogni passo, in cui ogni nodo è etichettato con l'hash dei suoi nodi figli. Tale struttura di dati è stata proposta da Merkle nel 1979 [15], per permettere una verifica efficiente e sicura dei contenuti in essa presenti. La struttura ad albero permette la verifica dell'esattezza delle informazioni contenute in un blocco attraverso una quantità di dati proporzionale al logaritmo del numero di nodi dell'albero. Come si può osservare nella figura 1.3, gli hash di tutti i blocchi, presi da una lista, vengono accoppiati ogni volta fino ad ottenere un solo hash, chiamato *top hash*. Per verificare l'integrità di un solo blocco è quindi necessario considerare un sottoinsieme dell'albero. Se ad esempio si vuole verificare l'integrità del blocco 1, si devono considerare i nodi *Hash 0-0*, *Hash 0-1*, *Hash 0*, *Hash 1* e *Top Hash*, ma non i nodi *Hash 1-0* e *Hash 1-1*. In particolare, nel sistema Bitcoin, ogni foglia dell'albero è costituita da una doppia funzione hash SHA-256[9].

Struttura del block header

Le informazioni contenute all'interno di un blocco sono illustrate in Figura 1.4.

- `index`: è il numero del blocco nella catena, il valore `index=0` è riservato per il blocco di genesi;

[9] Con il termine SHA (acronimo dell'inglese Secure Hash Algorithm) si indica una famiglia di cinque diverse funzioni crittografiche di hash sviluppate a partire dal 1993 dalla National Security Agency (NSA) e pubblicate dal NIST come standard federale dal governo degli USA **??**.

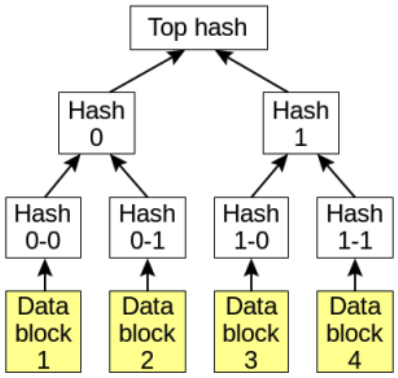

Figura 1.3: Esempio di albero di Merkle

Figura 1.4: Informazioni contenute nel block header

- `timestamp`: è l'instante di tempo nel momento in cui il blocco è stato aggiunto alla blockchain;
- `merkle root`: è l'hash della radice dell'albero di Merkle di tutte le transazioni appartenenti al blocco;
- `difficulty`: è il livello di difficoltà per validare un blocco;
- `previousHash`: è un puntatore al blocco precedente, il quale è il risultato della funzione di hash del blocco precedente, come viene illustrato in Figura 1.5;
- `hash`: è il risultato della funzione crittografica di hash di tutte le informazioni descritte precedentemente.

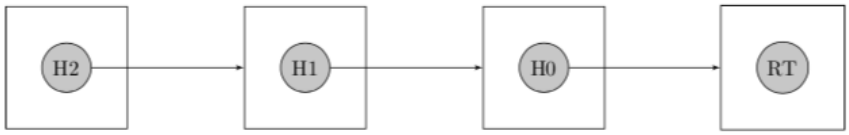

Figura 1.5: Struttura a catena con i puntatori di ogni blocco collegati al blocco precedente

Struttura del block body

Il block body è composto dalle informazioni relative alle transazioni, le quali vengono salvate all'interno del blocco attraverso una struttura ad albero di Merkle. Come descritto in precedenza, un Merkle tree è una struttura ad albero costruita creando l'hash di coppie di dati (le foglie), che vengono a loro volta accoppiate e viene creato un ulteriore hash, finché non si avrà un unico hash rimanente, chiamato *Merkle root*, la radice. L'hash di tale radice viene salvato nel *block header*. In Bitcoin, le foglie rappresentano le transazioni che vengono inserite in un singolo blocco.

Tale struttura permette di verificare un gruppo di transazioni, come illustrato in Figura 1.6.

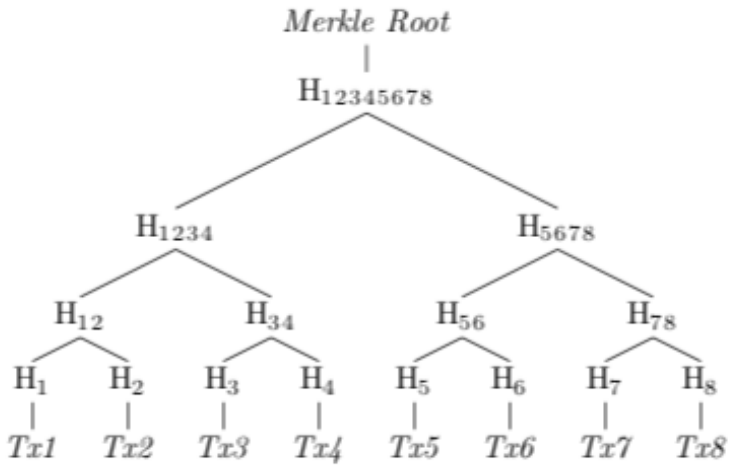

Figura 1.6: Block body: una struttura ad albero di Merkle

Processo di aggiunta dei blocchi

Il processo di aggiunta di un blocco alla blockchain è il seguente [16]:

1. le nuove transazioni e/o i nuovi blocchi vengono inviati ai nodi della blockchain;
2. ogni nodo salva temporaneamente in un area di buffer le transazioni e i blocchi ricevuti;
3. ogni nodo processa e analizza i blocchi seguendo il loro grado di priorità;
4. ogni nodo processa le nuove transazioni, dichiarandole valide o invalide ad essere autorizzate;
5. ogni nodo inserisce le transazioni valide in un Merkle tree e inizia a cercare un nuovo blocco per risolvere l'*hash puzzle* (che verrà descritto nel paragrafo successivo);
6. quando un nodo riesce a risolvere l'*hash puzzle*, invia il nuovo blocco agli altri nodi;
7. ogni nodo processa il nuovo blocco, verificando l'esattezza della soluzione trovata al*hash puzzle* e verificando le transazioni contenute nel blocco;
8. ogni nodo aggiunge i blocchi validati alla propria blockchain locale;
9. se uno dei nuovi blocchi è invalido, il blocco viene eliminato e i nodi riprendono a processare le transazioni o a completare l'*hash puzzle* di un nuovo blocco;
10. se un nuovo blocco è valido, i nodi rimuovono le transazioni contenute nell'area di buffer e cominciano a processare le transazioni ed a creare un nuovo blocco;
11. se il blocco aggiunto alla blockchain locale risulta successivamente invalido, il blocco viene rimosso e le transazioni vengono inserire nell'area di buffer, per essere processate in un secondo momento;

12. il nodo che emette il blocco accettato da tutti gli altri nodi riceverà una ricompensa, che altro non è che le commissioni pagate per effettuare le transazioni contenute nel blocco.

Protezione dei blocchi

Dopo che un blocco è stato aggiunto alla blockchain, non può essere più modificato. Ad esempio, un soggetto può alterare i dati relativi alla transazione Tx'_3, modificando la somma di denaro che sta per essere inviata al suo wallet con la transazione Tx_3. Tuttavia, questo comportamento implica la generazione di un hash differente, come è possibile osservare in Figura 1.7. Come risultato, ogni hash di livello superiore, fino alla radice, dovrebbe essere modificato per poter creare un blocco valido. Questo però richiede troppa potenza computazionale per essere performato da un singolo agente.

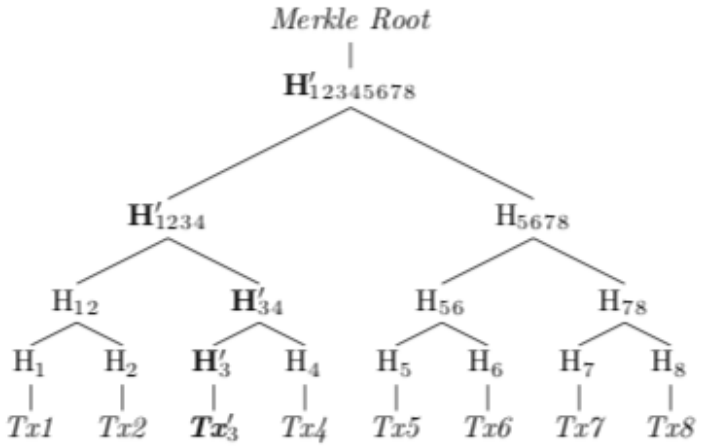

Figura 1.7: Risultato della modifica di una transazione su un blocco con struttura ad albero di Merkle

Un soggetto che volesse modificare le informazioni contenute in un blocco dovrebbe quindi modificare non solo l'hash della nuova transazione ma anche ogni hash di livello superiore, in modo tale da non far notare tale modifica agli altri nodi. La potenza computazionale richiesta ad un nodo per effettuare tale operazione è eccessiva,

comparata alla potenza di tutti gli altri membri della rete, i quali lavora in modo parallelo.

1.3.9 Hash puzzle

Oltre a permettere di comparare i dati, legarli l'un con l'altro e renderli sicuri, le funzioni di hash crittografico vengono utilizzate per creare "puzzle". Questi "puzzle" devono essere risolti da particolari nodi al fine di evitare frodi all'interno del sistema.

Gli hash puzzles presentano le seguenti caratteristiche:

- hanno bisogno di tempo per essere risolti;
- hanno un livello di difficoltà che può essere impostato;
- la verifica della soluzione dev'essere semplice e veloce.

In particolare, l'hash puzzle è una sfida con un determinato grado di difficoltà. Questo grado è espresso come un numero intero, che rappresenta il numero degli zeri che devono essere presenti nell'hash generato. Come detto in precedenza, la funzione che genera l'hash è unidirezionale, quindi gli stessi dati in ingresso genereranno lo stesso valore di hash in output; di conseguenza, per ottenere come output una stringa con un numero x di 0, la funzione di hash deve essere applicata non solo ai dati già presenti, ma anche ad un `nonce`[10]. Il `nonce` è un numero intero che viene iterativamente modificato finche l'hash della funzione non produce la soluzione al hash puzzle. Trovare tale soluzione richiede una notevole quantità di potenza computazionale, elettricità e tempo.

Questo processo di identificazione della soluzione al hash puzzle è chiamato "mining", il quale verrà spiegato più dettagliatamente nella Sezione 1.3.10.

[10] Il termine nonce indica un numero, generalmente casuale o pseudo-casuale, che ha un utilizzo unico. Nonce deriva infatti dall'espressione inglese for the nonce, che significa appunto "per l'occasione".

1.3.10 Mining

La grande quantità di potenza computazionale, energia e tempo richieste nel processo di "mining" comporta il fatto che non tutti i nodi del network hanno la possibilità di risolvere l'hash puzzle. I nodi che eseguono tale operazione sono chiamati "miners". Un nodo miner della blockchain ha il compito di trovare la soluzione dell'hash puzzle prima degli altri miners; il primo nodo che riesce a risolvere il puzzle viene ricompensato con le commissioni pagate dagli utenti per effettuare le transazioni e le nuove cryptovalute che vengono emesse tramite questo processo, ricompensandolo per il lavoro e lo sforzo computazionale fornito. I miners vengono quindi motivati da questi due incentivi monetari, come viene anche descritto da Narayanan et al. [17]. Tali ricompense sono uno strumento per accelerare e migliorare il processo di validazione delle transazioni e la conseguente creazione del blocco che le contiene. Tuttavia, la blockchain necessita anche di uno strumento punitivo nel caso in cui un nodo mantenga un comportamento scorreto nei confronti della rete. Le ricompense posso essere infatti requisite se un blocco accettato in passato viene dichiarato non valido o non utile; oppure, l'eliminazione della ricompensa può essere applicata se un nodo sta effettuando il mining di un blocco già aggiunto alla catena. Nel dettaglio, il mining comprende due processi: il *proof-of-work* e il *proof-of-stake*.

Proof of work

Il proof-of-work (PoW) è essenzialmento il processo di mining descritto nei paragrafi precedenti. E' un algoritmo che ha il principale obiettivo di dissuadere gli attacchi informatici come il DDoS[11], il quale cerca di esaurire le risorse di un sistema informatico inviando un elevatissimo numero di richieste false.

[11] Acronimo di Distributed Denial of Service (interruzione distribuita del servizio, in italiano.), l'attacco DDoS è un caso particolare di attacco DoS (semplicemente, denial-of-service). Lo scopo di un attacco DoS è quello di saturare le risorse (informatiche e di rete) di un sistema informatico che distribuisce diverse tipologie di servizio. Nell'ambito del networking, dunque, un attacco DoS punta a rendere irraggiungibile un sito o un server saturandone la banda di comunicazione.

Analizzando più nel dettaglio, la PoW è un metodo per dimostrare l'effettivo utilizzo di potenza computazionale, il mining, che dev'essere eseguito al fine di creare un nuovo gruppo di transazioni validate (il blocco) che vengono poi inserite nel *ledger* distribuito, cioè la blockchain.

L'attività di mining ha due scopi:

- verificare la legittimità delle transazioni, eliminando il rischio del *double-spending*[12];
- emettere nuova valuta digitale per ricompensare i miners per l'attività eseguita al punto precedente.

Tutti i miners presenti sula rete competono per trovare per primi la soluzione al hash puzzle: questo problema matematico può essere risolto solamente tramite un *brute force attack*, cioè provando iterativamente tutte le possibili soluzioni. Quando un miner riesce a trovare la soluzione esatta, la diffonde per tutta la rete e riceve la ricompensa prevista dal protocollo.

Da un punto di vista tecnico, il processo di mining è un operazione inversa di hash: è necessario determinare un numero (il nonce) per cui l'hash crittografico risultante sia inferiore ad una determinata soglia. Tale *threshold*, che rappresenta la difficoltà, è ciò che determina la natura competitiva del mining: più potenza computazionale viene aggiunta al network, più verrà aumentata questa soglia, incrementando così anche il numero medio di operazioni necessarie per la creazione di un nuovo blocco. Questo metodo aumenta inoltre il costo per la creazione di un blocco, spingendo i miners a migliorare l'efficienza dei loro sistemi di mining per mantenere un bilancio economico positivo.

[12] Il double-spending è un potenziale difetto in uno schema di cassa digitale in cui uno stesso singolo token digitale può essere speso più di una volta presso venditori diversi. Questo è possibile perché un token digitale è costituito da un file che può essere duplicato o falsificato. Come nel caso del denaro contraffatto, tale doppia spesa porta all'inflazione creando una nuova quantità di valuta fraudolenta che in precedenza non esisteva.

Proof of stake

La proof of stake (PoS) è un metodo differente per la validazione delle transazioni e la loro registrazione nella blockchain. Lo scopo rimane lo stesso della PoW, ma il processo per raggiungere tale obiettivo è differente.

Diversamente dalla PoW, dove l'algoritmo ricompensa i miners che riescono a risolvere un problema matematico, con lo scopo di validare le transazione e creare nuovi blocchi, nella PoS, il creatore di un nuovo blocco viene scelto deterministicamente, in base alla quantità di denaro fornita al sistema, chiamata *stake*. Prima della creazione di ogni blocco, i nodi hanno la possibilità di "bloccare" una determinata quantità di denaro, che funzionerà da garanzia per la validazione delle transazioni contenute nel blocco; in questo modo, i nodi vengono incentivati ad analizzare e validare le transazioni, eliminando ogni transazione errata, in quanto porterebbe alla totale perdita dello *stake* bloccato per quel determinato blocco. Il metodo di scelta del nodo che andrà a creare il blocco, e conseguentemente ricevere la ricompensa, differisce da blockchain a blockchain, ma generalmente viene scelto in maniera pseudo-casuale (i nodi che bloccheranno uno stake più alto avranno una maggiore probabilità di essere scelti).

Capitolo 2

Ethereum: Smart Contracts e piattaforma per DApps

Lo scopo di questo capitolo è quello di fornire un analisi di Ethereum, la piattaforma utilizzata per lo sviluppo di At0m, lo strumento d'investimento sviluppato per questa tesi e che verrà presentato nei prossimi capitoli.

2.1 Definizioni generali

Nella documentazione ufficiale, i nomi "Ethereum platform" e "Ethereum protocol" si riferiscono entrambi alla blockchain "Ethereum" stessa.

Definizione 2.1. Ethereum è una piattaforma pubblica ed open-source basata sulla blockchain, la quale permette di creare ed interagire con smart contracts e DApps (abbreviativo di Decentralized Applications) senza correre il rischio di downtime, frodi e interferenze da terze parti. Questa piattaforma permette di sviluppare applicazioni per blockchain con diverse logiche di business, le quali agiscono in un ambiente

completamente *trustless*[1], sfruttando al contempo l'elevata interoperabilità della rete Ethereum.

Il codice su cui si basa Ethereum è interamente disponibile su GitHub all'indirizzo www.github.com/Ethereum, il quale è diviso in diverse repository; le ricerche su cui l'Ethereum foundation sta attualmente lavorando sono disponibili sul sito web www.ethresear.ch.
Ether (ETH) è una cryptovaluta generata dalla piattaforma Ethereum; è la valuta nativa che alimenta e permette al sistema di operare. Può essere infatti trasferita fra diversi accounts, può essere utilizzata per pagare le commissioni sulle transazioni ed è la ricompensa per i nodi mining che forniscono potenza computazionale alla rete. Al momento, nei prossimi sviluppi di Ethereum, è programmato un upgrade del protocollo previsto per il 2019-2021 **[Ethereum-roadmap]** che prevede la sostituzione del mining (e quindi l'algoritmo Proof of Work) con il meccanismo del Proof of Stake, che permetterà di ridurre i consumi e la potenza computazionale richiesta per validare i dati all'interno della blockchain. Ether, infine, può avere altri utilizzi: può essere una riserva di valore (utilizzata, ad esempio, come collaterale per i prestiti), come mezzo di scambio e come unità di conto (ad esempio, nei marketplace digitali).

2.2 Obiettivi di Ethereum

Il fondatore di Ethereum, Vitalik Buterin, ha descritto gli obiettivi prefissati dalla blockchain nell'Ethereum whitepaper [18]:

> "What Ethereum intends to provide is a blockchain with a built-in fully fledged Turing-complete programming language that can be used to create ”contracts” that can be used to encode arbitrary state transition functions,

[1] Cioè un ambiente disegnato in modo tale che nessuno debba doversi "affidare" a nessun altro per far si che il sistema funzioni correttamente.

> allowing users to create any of the systems described above, as well as many others that we have not yet imagined, simply by writing up the logic in a few lines of code."

Ciò che intende fornire Ethereum è quindi una blockchain con un linguaggio di programmazione Turing-completo[2] integrato, il quale può essere utilizzato per creare dei "contratti" che possono essere utilizzati per codificare, tramite poche righe di codice, delle funzioni che permettano di passare da uno stato arbitrario all'altro, consentendo così di creare dei sistemi che, ad esempio, hanno la possibilità di trasferire automaticamente assets digitali secondo delle regole pre-definite.

2.3 Caratteristiche principali dell'Ethereum protocol

L'Ethereum protocol è volutamente focalizzato sulla costruzioni di applicazioni decentralizzate tramite l'utilizzo di smart contracts. Vitalik Buterin definisce cinque importanti principi su cui l'Ethereum protocol si basa [18]:

2.3.1 Semplicità

Mantenere la tecnologia il più semplice possibile significa aiutare i nuovi sviluppatori a creare applicazioni decentralizzate, senza dover essere a conoscenza di tutte le specifiche del protocollo; la semplicità è una caratteristica molto importante per Ethereum. Il white paper conferma tale importanza, affermando che ogni ottimizzazione che aggiunge complessità non verrà implementata, anche se tale ottimizzazione può portare ad un sostanziale beneficio:

> "The Ethereum protocol should be as simple as practical, but it may be necessary to have quite a high level of complexity, for instance to scale,

[2] Un sistema Turing completo vuol dire un sistema per il quale noi possiamo scrivere un programma che risolverà ogni problema computazionale (sebbene con nessuna garanzia riguardo la memoria richiesta e il tempo di esecuzione).

> to internalize costs of storage, bandwidth and I/O, for security, privacy, transparency, etc. Where complexity is necessary, documentation should be as clear, concise and up-to-date as possible, so that someone completely unschooled in Ethereum can learn it and become an expert."

2.3.2 Astrazione

Il concetto di astrazione, in Ethereum, si riferisce alle numerose possibilità di sviluppo in questa tipologia di piattaforma. E' infatti una piattaforma universale completamente open-source, dove gli sviluppatori possono sviluppare ogni tipo di programma senza alcun tipo di limitazione. Gli sviluppatori hanno la possibilità di creare ogni tipologia di smart contract, transazione, asset finanziario, cryptovaluta e token:

> "A fundamental part of Ethereum's design philosophy is that Ethereum does not have "features". Instead, Ethereum provides an internal Turing-complete scripting language, which a programmer can use to construct any smart contract or transaction type that can be mathematically defined."

2.3.3 Modularità

Ethereum è stato architettato in maniera modulare, cioè che il protocollo stesso è diviso in sezioni separate che permettono alla piattaforma di continuare a funzionare in tutte le situazioni, anche, ad esempio, se vengono implementati degli aggiornamenti. Il white paper riporta un chiaro esempio di questa modularità:

> "Over the course of development, our goal is to create a program where if one was to make a small protocol modification in one place, the application stack would continue to function without any further modification. Innovations [...] should be, and are, implemented as separate, feature-complete libraries. This is so that even though they are used in Ethereum, even if Ethereum does

not require certain features, such features are still usable in other protocols as well. Ethereum development should be maximally done so as to benefit the entire cryptocurrency ecosystem, not just itself."

2.3.4 Flessibilità

I fondatori di Ethereum non lo hanno progettato come una struttura permanente, ma piuttosto una piattaforma flessibile, come è reso chiaro anche nel white paper:

"Although we will be extremely judicious about making modifications to high-level constructs [...]."

2.3.5 Assenza di discriminazione e censura

Come ogni altra blockchain pubblica, la resistenza alla censura è delle caratteristiche principali. Ogni tipologia di utilizzo della piattaforma è consentito e non ci sono restrizioni a nessuna categoria specifica; sarà l'architettura del sistema stesso a regolare gli abusi:

"All regulatory mechanisms in the protocol should be designed to directly regulate the harm and not attempt to oppose specific undesirable applications. A programmer can even run an infinite loop script on top of Ethereum for as long as they are willing to keep paying the per-computational-step transaction fee."

2.4 Prime applicazioni client su Ethereum

Le prime tipologie di applicazioni comparse sulla piattaforma di Ethereum sono due, i wallets e i full nodes.

I wallets sono dei nodi light[3] che utilizzano la piattaforma Ethereum per inviare e ricevere cryptovalute [19]. Un wallet è semplicemente un'applicazione hardware o un software che detiene le chiavi per la EVM, la quale verrà descritta nei parametri successivi. Queste chiavi corrispondono ad un account, il quale si riferisce ad un indirizzo. I full nodes sono CLI[4] che permettono di performare il set completo di funzioni disponibili sulla rete.

Eseguire un full node su Ethereum significa avere la possibilità di creare ed effettuare il deploy di smart contracts sulla rete. Per chiarificare le funzioni di wallets e full nodes, devono essere prima introdotti i concetti di accounts, indirizzi (address) e transazioni.

2.5 Accounts

Un account è un oggetto[5] composto da dati: è un record del ledger della blockchain, indicizzato per il proprio indirizzo (address) e contenente una serie di dati riguardo al proprio *state*, che in Ethereum indica tutte le informazioni riguardanti un determinato account [**Ethereum-merkle-tree2019**] (dati sui fondi disponibili o sulle transazioni effettuate, ad esempio). Questo concetto è collegato all'associazione fra un indirizzo ed una chiave pubblica appartenente ad un particolare utente; è il metodo tramite il quale l'utente ha accesso al proprio account. L'indirizzo è tecnicamente l'hash di una chiave pubblica [20]. Le applicazioni wallet generalmente richiedono una password per proteggere le chiavi (spesso tramite criptatura); la chiave privata dovrebbe essere custodita con molta cura, mantenendola segreta.

[3] Sono dei nodi che, diversamente da quelli "full", non devono essere eseguiti 24/7 e non hanno bisogno di effettuare il download dell'intera blockchain, ma permettono comunque di eseguire transazioni.

[4] Command Line Interfaces, cioè programmi eseguibili di cui l'interfaccia grafica è il terminale.

[5] Con oggetto, in informatica, si intende una generica regione di memoria allocata. In particolare nella programmazione orientata agli oggetti, l'oggetto è un esemplare (in inglese: instance, comunemente anche se impropriamente tradotto con istanza) di una classe, unico e separato da altri oggetti con i quali può tuttavia "comunicare".

Gli accounts di Ethereum sono rappresentati da un lungo indirizzo esadecimale, come il seguente:

`D3186212819D152b6d27ac88762B147F55`

L'indirizzo è derivato dagli ultimi 20 byte della chiave pubblica che controlla l'account. L'indirizzo è spesso indicato esplicitamente preponendo 0x all'indirizzo. Poiché ogni byte dell'indirizzo è rappresentato da 2 caratteri esadecimali, l'indirizzo, con anche il prefisso, è lungo 42 caratteri.

E' importante rendere chiaro che gli ETH (*ticker* di Ethereum, simbolo presente per la quotazione) in Ethereum non sono contenuti in nessuna macchina o applicazione specifica; il bilancio degli ether di un indirizzo, come anche gli ether inviati e ricevuti, può essere richiesto tramite una query ad ogni nodo che sta eseguendo un Ethereum full node o un wallet. Questo perché Ethereum è una blockchain pubblica e completamente trasparente. Ogni transazione effettuata sulla rete principale di Ethereum (la cosiddetta *mainnet*) può essere consultata su uno dei tanti siti BlockExplorer[6] presenti sul web (come `www.etherscan.io`). Anche se il computer dove il wallet è presente viene distrutto, sarà necessaria solamente la chiave privata e si potrà nuovamente accedere agli ether dell'account, grazie agli altri nodi della rete.

Dall'altro lato, se la chiave andasse smarrita, gli ether sarebbero perduti per sempre. Inoltre, se qualcuno entrasse in possesso di una chiave privata di un altro utente, avrebbe accesso al suo wallet e potrebbe ritirare tutti i fondi.

Le misure minime necessarie per la sicurezza del proprio wallet sono quindi:

- effettuare il backup della chiave privata;
- conservare la propria chiave privata in un luogo sicuro.

[6] Un BlockExplorer è fondamentalmente un motore di ricerca che consente agli utenti di cercare e analizzare facilmente le transazioni che hanno avuto luogo sulla blockchain di Ethereum.

Gli accounts sono classificati in due pricipali tipolgie [21]: `externally owned accounts (EOAs)`, cioè gli account controllati dagli utenti tramite le proprie chiavi private oppure tramite un wallet; questi hanno al loro interno un saldo per gli ether posseduti, possono inviare e ricevere transazioni (e anche attivare il codice contenuto negli smart contracts, come vedremo nei successivi paragrafi) e non hanno alcun codice a loro associato. La seconda tipologia sono i `contract accounts`, i quali contengono il codice nel quale è programmata la logica dello smart contract; non hanno alcun saldo di ether associato e vengono attivati tramite una transazione ricevuta da altri accounts o altri contratti.

Tutto ciò che avviene sulla blockchain viene attivato tramite delle transazioni generate da accounts *EOAs*. Ogni qual volta che un *contract account* riceve una transazione, il suo codice viene eseguito come indicato dai parametri di input inviati come parte della transazione.

2.6 Transazioni

Una transazione è una singola istruzione firmata crittograficamente creata da un soggetto esterno e registrata all'interno di un blocco della rete Ethereum. Un umano è il grado più esterno per un soggetto; altrimenti, ulteriori strumenti software possono essere utilizzati per la costruzione e per la diffusione delle transazioni.

Sono presenti due tipologie di transazioni: quelle che risultano in *message calls*, cioè nell'invocazione di una funzione della blockchain, e quelle che risultano nella creazione di nuovi accounts associati a del codice, conosciuti formalmente come "*contract creation*". Entrambe le tipologie hanno in comune una serie di campi, discussi ampiamente nell'Ethereum Yellow Paper, la documentazione più affidabile riguardo alle transazione per la rete Ethereum [21].

I campi comuni più importanti sono:

- `nonce`: un numero unico che permette di bloccare la ripetizione della transazione nei blocchi successivi (il cosiddetto *replay attack*)[7];
- `gasPrice`: l'ammontare di ether che deve essere pagato per ogni unità di gas utilizzata per ricompensare lo sforzo computazionale utilizzato per eseguire la transazione;
- `gasLimit`: l'ammontare massimo di gas che dovrebbe essere pagato per permettere alla transazione di essere eseguita; è il budget che viene stabilito per la transazione;
- `to`: è l'indirizzo di destinazione della transazione; per la *contract creation* questo campo rimarrà vuoto;
- `value`: l'ammontare di ether che vengono inviati in queta transazione;
- `v, r, s`: sono dei valori necessari, da un punto di vista tecnico, per la firma della transazione e vengono utilizzati per identificare il mittente della transazione;

Inoltre, per una transazione *contract creation*, è presente anche il campo `init`, il quale specifica il codice deployato sulla Ethereum Virtual Machine per l'inizializzazione di una procedura. Al contrario, in una *message call*, è presente il campo `data`, il quale contiene tutti i dati di input della transazione. Uno degli aspetti fondamentali delle transazioni, e anche per l'arbitraggio (che verrà trattato nel prossimo capitolo), è il gas.

[7]Nell'ambito della sicurezza informatica il replay attack è una forma di attacco di rete che consiste nell'impossessarsi di una credenziale di autenticazione comunicata da un host ad un altro, e riproporla successivamente simulando l'identità dell'emittente (in questo caso una transazione.)

2.7 Gas

Uno dei problemi principali in un sistema a libero accesso come Ethereum è il cosiddetto *halting problem*[8]. La soluzione per evitare che si creassero problematiche sulla piattaforma è quella di addebitare una commissione per ogni step computazionale, il *gas*. Il gas è l'unità di costo della rete [21]. Fondamentalmente in Ethereum, gli ether possono essere liberamente convertiti in gas, e viceversa. Il gas non esiste al di fuori del motore di calcolo interno di Ethereum; il suo prezzo è espresso in gwei[9] dal mittente della transazione e i miners sono liberi di scegliere fra le transazioni con un *gas price* elevato e quindi una commissione maggiore. La rete Ethereum necessita quindi il gas per evitare problemi di abuso del network: in questo modo, ogni operazione effettuata in questo ambiente è soggetta a commissioni. Il prezzo è specificato in unità di gas; quindi, ogni frammento di calcolo programmabile ha un costo universalmente accettato in termini di gas, che comprende la creazione di contratti, l'esecuzione di chiamate di messaggi, l'utilizzo e l'accesso all'archiviazione dell'account e l'esecuzione di operazioni. E' importante tenere a mente che il gas non esiste al di fuori dell'esecuzione di una transazione. Ogni transazione ha un particolare campo, in cui è specificato l'ammontare di gas associato ad essa: il `gasLimit`. Questa è la quantità di gas che verrà implicitamente detratta dal saldo dell'account del mittente. L'acquisto avviene al `gasPrice` specificato nella transazione dal mittente; ovviamente, la transazione verrà considerata invalida se il saldo dell'account non è sufficiente per acquistare il gas. Il parametro `gasLimit` infatti agisce come una soglia: se verrà utilizzato meno gas rispetto a quello che era stato impostato, il resto verrà restituito all'account del mittente. In questo modo, per accounts a cui è associato del codice affidabile, un `gasLimit` relativamente alto potrebbe essere settato per permettere la sua continua esecuzione. In generale, gli ether

[8] Il problema della terminazione (dall'inglese Halting problem, tradotto anche con problema dell'arresto o problema della fermata) chiede se sia sempre possibile, descritto un algoritmo e un determinato input finito, stabilire se l'algoritmo in questione termina o continua la sua esecuzione all'infinito.

[9] GigaWei (o anche gwei), è un'altra denominazione di ether.

utilizzati per acquistare gas che non verrà restituito (per una transazione non valida), vengono inviati all'indirizzo del miner che ha validato il blocco. E' possibile specificare ogni livello di `gasPrice` per una transazione; tuttavia, i miners sono liberi di ignorare le transazione con un livello troppo basso. Un `gasPrice` più alto è un costo maggiore per il mittente, ma l'incentivo economico maggiore dato ai miners porterà ad avere una maggiore probabilità di essere inclusi nel blocco e validare la transazione. I miners, in generale, rendono pubblico il minimo `gasPrice` al quale sono disposti a validare le transazioni, e i loro mittenti sono liberi di scegliere qualsiasi livello di prezzo per il gas. Finché ci sarà una distribuzione pesata del prezzo minimo del gas, le transazioni avranno necessariamente un trade-off fra ridurre il costo del gas e massimizzare la probabilità che la transazione venga "minata" in maniera tempestiva. Questo vale per le transazioni ordinarie, ma è ancor più importante quando viene introdotto il concetto di arbitraggio, come vedremo nei prossimi capitoli. Una gestione ponderata del gas può portare ad una riduzione dei costi relativi alle commissioni permettendo un ritorno più elevato.

2.7.1 Unità di gas necessarie per le operazioni

Per chiarificare tale concetto, il l'*EVM (Ethereum Virtual Machine) code* e *EVM assembly* devono essere definiti:

- **EVM code** è un bytecode[10] che può essere eseguito nativamente dalla EVM; è il linguaggio in cui viene compilato uno smart contract.
- **EVM assembly** è la versione *human-readble* del EVM code

Le transazioni devono fornire abbastanza gas per ripagare lo sforzo computazionale e l'archiviazione. Ad esempio, ogni volta che una transazione viene eseguita, la somma-

[10] Il bytecode è un linguaggio intermedio più astratto tra il linguaggio macchina e il linguaggio di programmazione, usato per descrivere le operazioni che costituiscono un programma.

toria del costo delle sue operazioni ha un costo per il network pari al prezzo di 21000 unità di gas, in accordo ai costi delle operazioni EVM assembly [21] (vedi Tabella 2.1).

Tabella 2.1: Operazioni più significative della EVM

Value	Name	Gas used	Description
0x00	STOP	0	Halt the execution
0x01	ADD	3	Addition operation
0x02	MUL	5	Multiplication operation
0x06	MOD	5	Modulo remained operation
0x08	ADDMOD	8	Modulo addition operation
0x16	AND	3	Bitwise AND operation
0x20	SHA3	30+6*word	Keccak-256 hash
0x31	BALANCE	400	Get the balance of the given account
0x56	JUMP	8	After the program counter
0x54	SLOAD	200	Paid for a SLOAD operation
0x55	SSTORE	20000	To non-zero from 0
0x55	SSTORE	5000	Remains unchanged or set to 0
0xF0	CREATE	32000	Create new account with associated code

2.8 Valuta

Come già accennato, nella rete Ethereum è presente una propria valuta, *ether*, la quale ha sia lo scopo di fornire un layer primario di liquidità per uno scambio efficiente di differenti tipologie di assets digitali sia, come trattato nel paragrafo precedente, per pagare le commissioni sulle transazioni. L'unità minima di ether sono i *wei*, che quindi è l'unità per misurare tutti i valori decimali della valuta. In particolare, 1 ether corrisponde a 10^{18} wei. Nella Tabella 4.1 sono catalogate tutte le sottodenominazioni di ether, comparate anche al dollaro:

Tabella 2.2: Sottodenominazioni di Ether (prezzo rilevato 20/06/2019

Moltiplicatore	Nome
10^{18}	Wei
10^{15}	Kwei
10^{12}	Mwei
10^{9}	Gwei
10^{6}	Szabo
10^{3}	Finney
1	Ether
10^{-3}	Kether
10^{-6}	Mether
10^{-9}	Gether
10^{-12}	Tether
268.60	USD

Tali sottodenominazioni sono simili al concetto di "dollaro" e "centesimo" o "bitcoin" e "satoshi". Gwei è attualmente l'unità utilizzata per il prezzo del gas.

2.8.1 Prezzo del gas in Gwei

Come detto precedentemente, il prezzo del gas viene misurato in Gwei. Il prezzo del gas può essere può essere impostato senza limiti dal mittente di una transazione; più alto è il livello di prezzo del gas impostato dal mittente e più veloce sarà la transazione che verrà processata dal miner. Di solito, per i mittenti sono definiti tre prezzi:

Tabella 2.3: Fasce di prezzo per le transazioni

Nome	Prezzo (in gwei)	Tempo di esecuzione
safe low	1	< 30 min
standard	2	< 5 min
fast	18	< 2 min

Source: www.ethgasstation.info

2.9 EVM: Ethereum Virtual Machine

La EVM può essere vista come un computer mondiale che chiunque può utilizzare, al costo di una minima commissione pagabile in ether. La EVM è un singolo computer globale a 256-bit in cui tutte le transazioni vengono archiviate in ogni nodo della rete, e vengono eseguite in una relativa sincronia. Può essere definita, inoltre, una macchina virtuale globalmente accessibile, composta da una moltitudine di computers [20].

Questo computer, al quale chiunque colleghi un nodo o un wallet può accedere, rende semplice trasferire grandi quantità arbitrarie di valore in pochi istanti. Anche se chiunque può utilizzare questa macchina virtuale globale, nessuno può emettere denaro contraffatto al suo interno o muovere fondi senza l'autorizzazione. Sembra inutile avere l'intera EVM, tutti quei nodi, i quali replicando le stesse transazioni e conservano servilmente lo stesso stato tra migliaia di singoli computer, ma è importante avere una base adeguata per il confronto su come i servizi finanziari IT funzionano oggi.

L'EVM è un esempio di semplicità ed efficienza al confronto. Ancora più importante, tutto quel "lavoro" non è per nulla; In effetti, è la prova di questo "lavoro" che protegge la rete (Proof of Work). Ormai, l'EVM è una macchina virtuale generalizzata, sicura e priva di proprietà che offre funzionalità economiche simili a Fedwire con implementate ulteriori funzioni.

La EMV è una macchina singleton[11] di transazione con stato condiviso. In informatica, ciò significa che si comporta come un gigantesco oggetto di dati, piuttosto che ciò che è: una rete di macchine separate, a loro volta singleton, in costante comunicazione. Per la prospettiva di uno sviluppatore di software, la EVM è anche un ambiente runtime per piccoli programmi che possono essere eseguiti dalla rete (smart contracts). L'EVM può eseguire arbitrariamente questi programmi per computer, scritti principalmente in

[11] Il singleton, nella programmazione ad oggetti, è un design pattern ed che ha lo scopo di garantire che di una determinata classe venga creata una e una sola istanza, oltre a fornire un punto di accesso globale a tale istanza.

Solidity, che verrà approfondito nei paragrafi successivi. Questi programmi, dato un particolare input, producono sempre, allo stesso modo, lo stesso output, con le stesse variazioni dello stato sottostante. Per questo motivo, i programmi Solidity sono completamente deterministici; inoltre, è garantito eseguirli a condizione che l'utente abbia pagato abbastanza per le transazioni.

I programmi scritti in Solidity sono in grado di esprimere tutte i task eseguibili dai computer, rendendoli teoricamente Turing completi. Ciò significa che l'intera rete, quindi ogni nodo, esegue qualunque programma presente ed attivo sulla piattaforma. Quando un utente carica uno smart contract attraverso il proprio nodo Ethereum, questo viene salvato nell'ultimo blocco e propagato nella rete, dove viene archiviato su ogni altro nodo nell'EVM in modo tale da eseguire lo stesso codice, come parte del protocollo di elaborazione dei blocchi. I nodi analizzano il blocco che sta per essere processato ed eseguono qualsiasi codice inserito tra le transazioni; ogni nodo lo fa in modo indipendente. Tale operazione non è altamente parallelizzata, ma altamente ridondante. L'EVM è quindi una *state machine*. Le *state machines* sono macchine provviste di memoria che sono perennemente attive. Come *state machine*, l'EVM ha una cronologia costante di tutte le transazioni all'interno dei propri banchi di memoria, in cui è possibile risalire alla prima transazione. Lo stato di un computer è il risultato specifico di ogni singolo cambio di stato (come una transazione finanziaria, dove lo stato di un determinato account cambia aumentandone il saldo se, ad esempio, ha ricevuto un determinato ammontare di tokens) che si è verificato all'interno della macchina. L'ultima versione dello stato della macchina è la "verità" canonica della macchina sulla realtà così com'è in quell'istante. In Ethereum questa verità riguarda i saldi contabili e la serie di transazioni eseguite. Le transazioni, quindi, rappresentano una tipologia di narrativa della macchina, una gamma di calcoli validi tra uno stato e l'altro. Come dice lo Yellow Paper (dove è presente la maggior parte della documentazione prettamente informatica) di Ethereum di Gavin Wood [21]:

> "Ethereum, taken as a whole, can be viewed as a transaction-based state machine: we begin with a genesis state and incrementally execute transactions to morph it into some final state. It is this final state which we accept as the canonical "version" of the world of Ethereum. The state can include such information as account balances, reputations, trust arrangements, data pertaining to information of the physical world; in short, anything that can currently be represented by a computer is admissible. Transactions thus represent a valid arc between two states; the "valid" part is important, there exist far more invalid state changes than valid state changes. Invalid state changes might, e.g., be things such as reducing an account balance without an equal and opposite increase elsewhere. A valid state transition is one which comes about through a transaction."

La EVM è da considerare una delle macchine più affidabili che fra le reti globali attuali. La sua esecuzione è *deterministica*, in quanto l'output che viene prodotto (il blocco contenente le informazioni relative alle transazioni) da un preciso input (le transazioni proposte inizialmente ai miners) è sempre lo stesso. Per ogni istruzione eseguita dall'EVM, deve esserci un costo associato; tale meccanismo permette di scoraggiare l'implementazione di contratti inutili (esistono anche reti test). Ogni volta che un'istruzione viene eseguita, un contatore interno tiene traccia delle spese sostenute, che vengono addebitate a chi le sta eseguendo. Ogni volta che un mittente avvia una transazione, il suo wallet riserva sempre una piccola parte per pagare le commissioni. Dopo che una transazione è stata trasmessa alla rete da un determinato nodo, la rete diffonde la transazione in modo che tutti i nodi possano includerla nell'ultimo blocco.

2.9.1 Implementazioni

Le attuali implementazioni della EVM live sul main nettwork sono [21]:

- `go-Ethereum`, un popolare client[12] per Ethereum con la propria implementazione della EVM (core/vm directory);
- `Parity`, scritto in Rust, un altro popolare client per Ethereum con la prorpria implementazione della EVM (ethcore directory);
- `PyEthereum`, un client ormai deprecato (Ethereum/vm.py);
- `Py-EVM`, un implementazione per Python strutturate per essere altamente configurabile e modulare, conforme alla Ethereum test suite, è ancora in via di sviluppo;
- `EthereumJ`, scritto in Java, è un ulteriore client con la sua implementazione della EVM.

2.9.2 Linguaggi di programmazione

La seguente lista riporta i differenti linguaggio di programmazione compilabili nella EVM:

- `Solidity`, il linguaggio più popolare per scrivere smart contracts in Ethereum
- `Vyper`, un linguaggio simile a Python e focalizzato sulla sicurezza;
- `Flint`, un ulteriore linguaggio con diverse caratteristiche focalizzate sulla sicurezza;
- `Bamboo` (sperimentale), un linguaggio senza loops ma con un'invocazione esplicita al constructor alla fine di ogni chiamata.

[12] Un client, in informatica, indica genericamente un qualunque componente che accede ai servizi o alle risorse di un'altra componente detta server. In questo contesto si può quindi parlare di client riferendosi all'hardware oppure al software. Esso fa parte dunque dell'architettura logica di rete detta client-server.

2.9.3 Debuggers

La seguente lista riporta i principali debuggers disponibili agli sviluppatori Ethereum:

- `Remix`, una IDE contenente un debugger per il codice della EVM;
- `debug_traceTransaction`, è un metodo che traccia le informazioni fornite da go-Ethereum.

2.10 Smart contracts

Il concetto di smart contract è stato introdotto da Szabo e Miller nel 1997 [1]. Negli anni 90, divenne chiaro che un trasposizione algoritmica degli accordi fra due o più parti potesse diventare uno strumento utile nella cooperazione fra individui. Sebbene non sia stato proposto alcuno strumento specifico per implementare tale sistema, fu ipotizzato che in futuro tale implementazione potesse pesantemente influenzare l'ordinamento giuridico in materia contrattuale. Sotto questo aspetto, Ethereum può essere visto come una generale implementazione di tali sistema basato su una *crypto-regolamentazione*.

2.10.1 Necessità di introduzione degli smart contracts

Le applicazioni presenti sulla EVM sono indicate anche con il nome di smart contracts [20]. In contesto giuridico, il termine "contratto" si riferisce a una tipologia specifica: i contratti finanziari. Tali contratti sono accordi per comprare o vendere un bene o un servizio, di solito ad un prezzo specifico. Nel contesto di Ethereum, gli smart contracts sono accordi fra differenti accounts, nei quali vengono definite determinate condizioni che, nel momento in cui vengono soddisfatte, attivano il trasferimento di assets digitali. Tali contratti vengono definiti "smart" in quanto vengono eseguiti da una macchina e le risorse (ether e altri token) vengono spostate automaticamente. Questi contratti posso-

no essere eseguiti anche centinaia di anni dopo che sono stati scritti, supponendo che la rete sia ancora in funzione, e anche se una moltitudine di utenti malintenzionati cercano di interferire. La EVM è un ambiente totalmente in modalità *sandbox*[13] e privo di ogni interferenza e, al momento, isolato da altre reti; queste condizioni rendono impossibile per una delle due parti sottrarsi a quanto riportato nel contratto: quando le condizioni vengono soddisfatte, le operazioni definite nel contratto vengono automaticamente eseguite.

In termini pratici, questo è dovuto al fatto che gli smart contracts sono autorizzati ad essere dei garanti per gli assets oggetto del contratto e a muoverli quando i termini del contratto sono soddisfatti. Gli smart contrats possono essere anche visti come delle "*s-box*" crittografiche; in crittografia, le *s-box* (o *Substitution-box*, letteralmente "scatole di sostituzione") sono dei componenti base degli algoritmi a chiave simmetrica[14]. Tali s-box contengono valore che viene "sbloccato" solo se vengono soddisfatte determinate condizioni.

2.10.2 Solidity: un linguaggio di programmazione per smart contracts

Solidity è un nuovo linguaggio di programmazione high level[15]; viene utilizzato per scrivere programmi chiamati smart contracts. Solidity è un linguaggio "*statically typed*", nel quale la tipologia di ogni variabile (che può essere *state* e *local*) dev'essere specificata (o almeno già dichiarata) nel momento in cui lo si compila [22].

[13] In ambito informatico, una *sandbox* identifica un ambiente di test, di prova, spesso slegato dal normale flusso di ambienti predisposti per lo sviluppo e il test delle applicazioni.

[14] Con crittografia simmetrica, o crittografia a chiave privata, si intende una tecnica di cifratura. Rappresenta un metodo semplice per cifrare testo in chiaro dove la chiave di crittazione è la stessa chiave di decrittazione, rendendo l'algoritmo molto performante e semplice da implementare. Tuttavia presuppone che le due parti siano già in possesso delle chiavi, richiesta che non rende possibile uno scambio di chiavi con questo genere di algoritmi. Lo scambio avviene attraverso algoritmi a chiave asimmetrica o pubblica (algoritmi utilizzati anche nella blockchain), generalmente più complessi sia da implementare che da eseguire ma che permettono questo scambio in modo sicuro. Dopodiché la comunicazione verrà crittata usando solo algoritmi a chiave simmetrica per garantire una comunicazione sicura ma veloce.

[15] Un linguaggio di programmazione ad alto livello (high level), in informatica, è un linguaggio di programmazione caratterizzato da una significativa astrazione dai dettagli del funzionamento di un calcolatore e dalle caratteristiche del linguaggio macchina, rendendolo più semplice e diretto da utilizzare.

Tale linguaggio è stato influenzato da C++, Python e JavaScript, ed è stato progettato per poter interagire con la EVM. Solidity è statically tiped, supporta l'ereditarietà[16], librerie e altre tipologie complesse di variabili definite dall'utente, tra le varie funzionalità. Questo nuovo linguaggio usa fondamentalmente un insieme di convenzioni che provengono dal mondo del networking, dell'assembly[17], dello sviluppo web e della sicurezza informatica.

2.11 Tokens

La piattaforma Ethereum consente inoltre di creare degli smart contracts con funzione di tokens, cioè delle valute digitali che possono essere utilizzate e scambiate solamente attraverso la blockchain Ethereum. Questo sta a significare che chiunque sviluppi un'applicazione o un servizio basato su Ethereum, può anche emettere un token proprietario. Questi tokens si dividono in due categorie principali:

- *asset tokens*, i quali rappresentano il diritto di proprietà di un determinato bene (materiale o immateriale);
- *utility tokens*, garantiscono ai loro possessori l'accesso futuro a beni o servizi tramite un'infrastruttura basata sulla blockchain.

Da un punto di vista tecnico, nella rete Ethereum, questi tokens vengono definiti come *ERCs* (Ethereum Requests for Comments) e sono dei documenti tecnici utilizzati dagli sviluppatori di smart contracts; essi definiscono una serie di regole necessarie per l'implementazione dei tokens per l'ecosistema Ethereum. Questi documenti vengono solitamente creati dagli sviluppatori stessi, e includono una serie di specifiche

[16] In informatica l'ereditarietà è uno dei concetti fondamentali nel paradigma di programmazione a oggetti. Essa consiste in una relazione che il linguaggio di programmazione, o il programmatore stesso, stabilisce tra due classi.

[17] In informatica, Assembly è un linguaggio a basso livello per la programmazione di computer, vicino al linguaggio macchina. Rappresenta attraverso dei simboli le istruzioni in linguaggio macchina o le costanti numeriche utilizzate per programmare una specifica CPU.

riguardo al loro protocollo. Prima di diventare dei documenti standard, un ERC dev'essere revisionato, commentato ed accettato dalla comunità attraverso un *EIP* (Ethereum Improvement Proposal), cioè una proposta contenente tutte le informazioni tecniche.

Verrà ora analizzata la tipologia più diffusa di tokens: l'ERC-20.

2.11.1 ERC-20

La tipologia di tokens ERC-20 permette l'implementazione di un'API standard per garantire l'interoperabilità tra token. Offre funzionalità di base per trasferire token, ottenere i saldi dei conti, ottenere la quantità totale di token in circolazione e consentire l'approvazione dei token. Per definire un token ERC20 è necessario:

- l'indirizzo del contratto;
- il numero di token disponibili.

Tuttavia, sono presenti ulteriori valori facoltativi per fornire ulteriori informazioni riguardo al token, come:

- nome;
- simbolo;
- decimali, cioè quante volte è possibile dividere il token. È possibile scegliere tra 0 e 18 valori decimali.

L'ERC-20 introduce due tipologie di eventi, `Transfer()`, attivato quando vengono trasferiti dei tokens e `Approve()`, utilizzato per ogni chiamata avvenuta con successo del metodo `approve()`. Un token può includere anche funzioni come `allowance()`, `approve()` e `transferFrom()` per offrire funzionalità avanzate e autorizzare altri indirizzi Ethereum ad utilizzare i token per conto dell'utente. Questo altro indirizzo di Ethereum potrebbe essere uno smart contract progettato per gestire tokens o un altro account.

Questa tipologia di tokens, soprattutto per la sua diffusione e per l'alto numero di tokens ERC-20 presenti nell'ecosistema Ethereum, viene utilizzata nell'arbitraggio atomico, concetto che verrà introdotto nei seguenti capitoli. Questo permette di avere numerose coppie di valute digitali contrattate in varie piattaforme, che portano ad avere diverse discrepanze di prezzo, permettendo agli arbitraggisti di ottenere un profitto riallineando i prezzi fra i diversi mercati.

2.12 DApps: applicazioni decentralizzate

Le DApps sono applicazioni, solitamente applicazioni Web, che vengono eseguite in un browser e interagiscono direttamente con gli smart contracts presenti sulla blockchain. Un'applicazione Web tradizionale generalmente utilizza un client Web che effettua delle richieste API ad un server che gestisce il back-end dell'applicazione. I dati che verrebbero archiviati in un database completamente di proprietà dell'applicazione e la sua logica verrebbe applicata solamente nel layer delle API. In una DApp, l'applicazione web non fa altro che leggere i dati direttamente dalla blockchain e inserisce i dati al suo interno tramite delle transazioni. La maggior parte della logica dell'applicazione viene applicata da uno smart contract presente sulla blockchain.

2.13 DExs: mercati decentralizzati

Come già trattato nella sezione 2.10, gli smart contracts sono programmi informatici eseguiti senza l'intervento dell'utente; l'esecuzione avviene automaticamente all'interno del sistema, il quale permette anche la verifica della corretta esecuzione di tali programmi. Gli smart contracts utilizzano la blockchain come infrastruttura sottostante per la loro esecuzione [21]. L'attuale applicazione più diffusa (in termini di utenti attivi giornalmente) degli smart contracts, per quanto riguarda la piattaforma Ethereum, sono i tokens, analizzati nella Sezione 2.11.

I DEXs, abbreviazione di *decentralized exchanges*, sono un'ulteriore tipologia di smart contracts, i quali permettono agli utenti di scambiare i sopracitati tokens in modo completamente *non-custodial*; ciò significa che tali piattaforme non custodiscono i fondi dei loro utenti in un singolo wallet o in un server centrale. Invece, gli utenti sono in possesso dei loro fondi e autorizzano lo smart contract del DEX ad accedere ai loro fondi per effettuare le operazioni di trading. Questo permette di eliminare il rischio, legato alla sicurezza della piattaforma, di custodire gli assets dei clienti in un singolo wallet; i fondi, infatti, vengono scambiati tramite il modello del peer-to-peer.

2.13.1 Mercati con *continuous-limit orderbook*

I mercati finanziari tradizionali, come il mercato azionario, quello delle *commodities*, o anche alcuni exchange di cryptovalute, condividono generalmente un comune e globalmente accettato design per la gestione dell'orderbook: il *continuous-limit orderbook*. Tale tipologia di orderbook consiste in una lista di ordini di vendita e di acquisto presenti in un mercato. I potenziali acquirenti inseriscono un *limit order*, che specifica il prezzo massimo al quale essi sono disposti ad acquistare un asset; analogalmente i venditori inseriscono dei *limit order* di vendita. Dopodiché un autorità centralizzata, in questo caso gli operatori di mercato, confrontano gli ordini dei venditori e degli acquirenti, eseguendo automaticamente le transazioni dove il prezzo di vendita di uno specifico ordine è minore del prezzo di acquisto di un ulteriore ordine. Gli ordini vengono quindi abbinati o inseriti nell'orderbook dall'operatore di mercato, il quale elabora tali ordini nell'ordine in cui vengono ricevuti. Non appena gli ordini coincidono, vengono eseguiti e i saldi degli account degli utenti coinvolti nell'operazione vengono modificati.

2.13.2 Tipologie di DEXs orderbook

Anche per alcuni DEXs il design utilizzato per l'orderbook è quello del *continuous-limit*; gli utenti mantengono i loro assets *on-chain*, quindi all'interno dei loro accounts, e

lo smart contract dell'exchange interpreta il ruolo degli operatori di mercato [23]. Una tipologia più radicale di design per l'orderbook, chiamata *automated market maker* [24], bypassa completamente il concetto di orderbook. Questi DExs consistono in uno smart contract che a sua volta detiene una riserva di token e/o ether. Gli orderbooks vengono solitamente mantenuti *off-chain*. In alcuni design, una delle due controparti seleziona un ordine direttamente dall'orderbook e lo invia allo smart contract con il corrispondente controrordine. Lo smart contract esegue la transazione ed elimina l'ordine presente sull'orderbook; questo approccio viene utilizzato da exchanges come Etherdelta [25] o alcune applicazioni di 0x [26] (per un analisi più dettagliata, si veda l'Appendice C). Si considerino due asset, A e B; lo smart contract permette all'utente di scambiare A e B in qualsiasi momento utilizzando le riserve presenti sulla piattaforma come controparte, ad un tasso prestabilito (deciso, solitamente, tramite un algoritmo pubblicamente visionabile). Se un utente acquista A usando B, il prezzo di A in termini di B offerto dallo smart contract per i trade successivi aumenta. Viceversa, se un utente vende A per acquistare B, il prezzo diminuisce. In questo modo, le singole parti possono effettuare degli scambi senza dover trovare una controparte che soddisfi il loro ordine. Di conseguenza, è necessario un arbitraggio costante su questa tipologia di piattaforme per poter mantenere i tassi di cambio allineati con i prezzi di mercato. Uniswap (www.uniswap.io) e Bancor (www.bancor.network) sono esempi di tali piattaforme.

Capitolo 3

Arbitraggio di prezzo tra valute

3.1 Definizione di arbitraggio

L'arbitraggio è generalmente definito come la capitalizzazione di una discrepanza dei prezzi di uno strumento finanziario, innescata dalla rottura della condizione di equilibrio del prezzo (*equilibrium pricing condition*) [27]. Succede spesso che l'arbitraggio venga raffigurato come un'operazione completamente senza rischi, in quanto che tutte le variabili decisionali sono conosciute prima ancora che l'operazione avvenga; ma nella sua applicazione reale, come vedremo nei paragrafi conclusivi di questo capitolo, le operazioni che compongono l'arbitraggio possono includere una componente di rischio dovuta da fattori esterni alla concezione teorica dell'arbitraggio. Il processo di arbitraggio ristabilisce l'equilibrio correggendo la domanda e l'offerta dello strumento finanziario preso in considerazione. Questo cambiamento di domanda e offerta provoca una variazione del prezzo in modo tale da ristabilire la condizione di equilibrio. Nel caso particolare del *foreign exchange market*, il mercato valutario internazionale, l'arbitraggio può essere definito come un'operazione simultanea di acquisto e rivendita di una valuta, con l'intento di generare profitto. Le opportunità profittevoli di arbitraggio sorgono sia nei mercati valutari *spot* che in quelli *forward*, entrambi a causa delle

differenze nelle quotazioni delle valute in un mercato finanziario rispetto ad un altro. Risulta quindi doveroso segnalare il fatto che nei mercati finanziari odierni, le opportunità di arbitraggio, come quelle appena descritte, si presentano in modo estremamente raro. E anche se queste finestre di arbitraggio riescono ad essere individuate, vengono sfruttate istantaneamente dagli arbitraggisti, fino a farle scomparire del tutto. É, comunque, importante approfondire lo studio di queste operazioni in quanto forniscono il meccanismo in cui l'equilibrio del prezzo è mantenuto tale.

3.2 Arbitraggio deterministico

L'arbitraggio deterministico è ogni tipo di strategia di arbitraggio dove il profitto può essere calcolato ex ante con assoluta certezza, con l'assunzione che tutte le transazioni possano essere eseguite simultaneamente, escludendo quindi la componente di rischio. Ciò significa che il profitto è conosciuto ancora prima che le transazioni vengano eseguite.
Thorp [28] descrive l'arbitraggio come l'acquisto di uno strumento sottovalutato e la simultanea vendita dello stesso strumento sovra-valutato. In quanto gli strumenti dovrebbero essere equivalenti (non necessariamente identici), questa transazione dovrebbe raggiungere un grado di correlazione pari a -1, mitigando quindi sia i rischi dei coefficienti beta (rischio sistematico) e alpha (rischio specifico) [29], portando quindi ad un profitto privo di rischi. Kuepper [30] invece stabilisce l'esistenza di un'opportunità di arbitraggio deterministico solamente quando una o più delle seguenti condizioni vengono violate:

- la compravendita di uno stesso strumento deve avvenire allo stesso prezzo in tutti i mercati;

- due strumenti finanziari con un identico cash-flow devono essere scambiati allo stesso prezzo;

- uno strumento di cui si conosce già il prezzo del *future* dev'essere attualmente scambiato al prezzo scontato del risk-free rate.

Dal momento che molti investitori sono alla ricerca di un profitto sicuro e senza rischi, l'arbitraggio deterministico è un'area altamente competitiva e che richiede l'accesso *real-time* ai prezzi degli strumenti finanziari, l'esecuzione istantanea delle operazioni e l'abilità di poter prendere in prestito fondi ad un tasso d'interesse basso [31]. Kuepper aggiunge inoltre che gli investitori retail [30] non possono essere competitivi quando si parla di arbitraggio deterministico, in quanto tale settore è completamente dominato dai *market-makers*. Per esempio, il valore della maggior parte degli strumenti finanziati è influenzato dai cambiamenti che la banca centrale effettua sui tassi d'interesse. Per poter ottenere un profitto dagli imminenti movimenti del mercato, un arbitraggista dovrebbe ricevere il più velocemente possibile quelle informazione, per poi processarle e agire di conseguenza.

3.3 Arbitraggio bilaterale

3.3.1 Arbitraggio bilaterale puro

Conosciuto anche con il nome di *two-point arbitrage* (o anche *spatial arbitrage*), le occasioni di arbitraggio bilaterale sorgono nel momento in cui il prezzo di due asset non è il medesimo in due diversi mercati finanziari. Sebbene l'arbitraggio bilaterale può essere effettuato sia nei mercati *spot* che in quelli *forward*, la seguente discussione verrà ristretta agli *spot markets*.
Si supponga l'esistenza di due distinti mercati finanziari, A e B, due valute, x e y, e che i costi di transazione (come ad esempio le commissioni d'intermediazione), la tassazione

e il *bid-ask spread* siano pari a zero. Se $S(x/y)_A$ è il prezzo *spot* tra x e y nel mercato A misurato come il prezzo (in termini di x) di un'unità di y, allora l'arbitraggio bilaterale potrà avvenire se:

$$S_A(x/y) \neq S_B(x/y) \tag{3.1}$$

Ciò significa che il prezzo fra le due valute presenta una differenza nel valore fra i due istituti finanziari in un determinato istante di tempo. Per semplificare la notazione, nei prossimi paragrafi, verrà omessa l'indicazione dell'unità di misura (x/y). Se le condizioni di arbitraggio rappresentate dalla relazione (3.1) vengono violate, un possibile scenario può essere

$$S_A > S_B \tag{3.2}$$

il quale indica che la valuta y ha un prezzo inferiore in B rispetto che in A (o, equivalentemente, che la valuta x sia meno costosa in A). L'arbitraggio bilaterale, in questo caso, si attuerebbe comprando y dove è più economico (in B), rivendendo poi dove è più costoso (in A). Il profitto per unità di x realizzato dall'operazione, π, è la differenza tra quanto ottenuto dalla vendita e quanto pagato per l'acquisto, o

$$\pi = S_A - S_B \tag{3.3}$$

La formula (3.1) mostra come l'arbitraggio influenza le forze della domanda e dell'offerta, e quindi anche i tassi di cambio in entrambi i mercati finanziari. Come la domanda cresce per y in B, S_B aumenta a sua volta, così come l'aumento dell'offerta di y cresce in A, S_A diminuisce, riducendo così il profitto dell'arbitraggio (come in Figura 3.1). Questo processo continua finché tale profitto diminuisce fino ad arrivare a zero ($\pi = 0$).

La condizione di assenza di arbitraggio viene quindi raggiunta nel momento in cui

$$S_A = S_B \tag{3.4}$$

Pertanto, ogni violazione della condizione rappresentata dall'equazione (3.4) innesca l'apertura di una finestra di arbitraggio profittevole. La condizione di assenza di arbitraggio può essere rappresentata, come riportato nella figura dai punti che compongono sulla "linea di non-arbitraggio" (Figura 3.2), la quale è una linea inclinata a 45° e che passa per l'origine. I punti che non risiedono su tale retta rappresentano la violazione della condizione di assenza di arbitraggio. L'operazione di arbitraggio causa un movimento verso questa linea sia per l'incremento di S_B, sia per la diminuzione di S_A, o, più verosimilmente, per entrambe le variazioni. I punti al di sotto della linea indicano la violazione nella forma di $S_A > S_B$. Nel seguente paragrafo verranno introdotte alcune complicazioni che avvicinano questa rappresentazione a ciò che succede nel mondo reale.

3.3.2 Arbitraggio bilaterale con commissioni

Assumiamo che i compratori e i venditori debbano pagare una commissione per l'intermediazione (*brokerage fees*) sulle transazioni della compravendita di valute. Si assuma inizialmente che queste commissioni siano fisse e indipendenti dalla dimensione delle transazioni. Si supponga quindi che un arbitraggista voglia ottenere del profitto comprando y in B e vendendolo in A. Nello scenario attuale, il profitto realizzato dall'operazione è

$$\pi = S_A - S_B - (\beta_A + \beta_B) \tag{3.5}$$

dove $\beta_A + \beta_B$ sono le commissioni di intermediazione rispettivamente nel mercato finanziario A e B. Per far sì che l'operazione di arbitraggio sia profittevole, la seguente

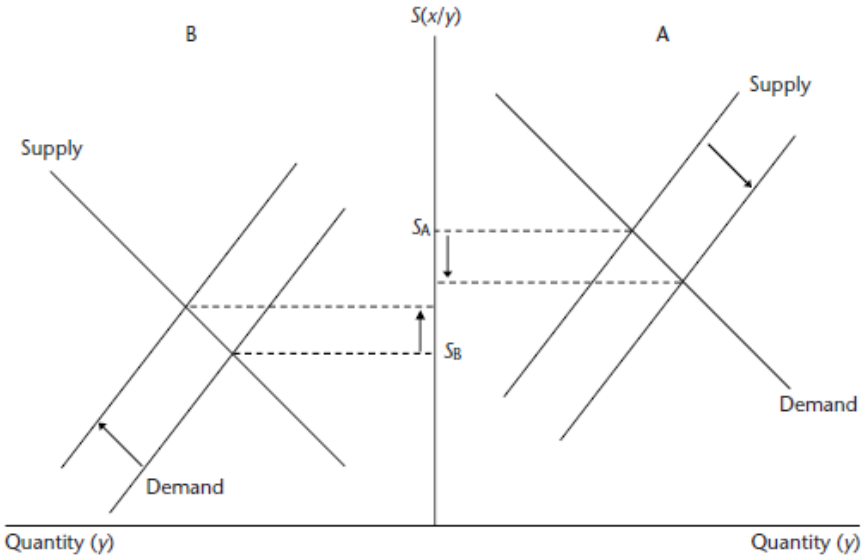

Figura 3.1: Effetto dell'arbitraggio bilaterale sulla domanda e sull'offerta

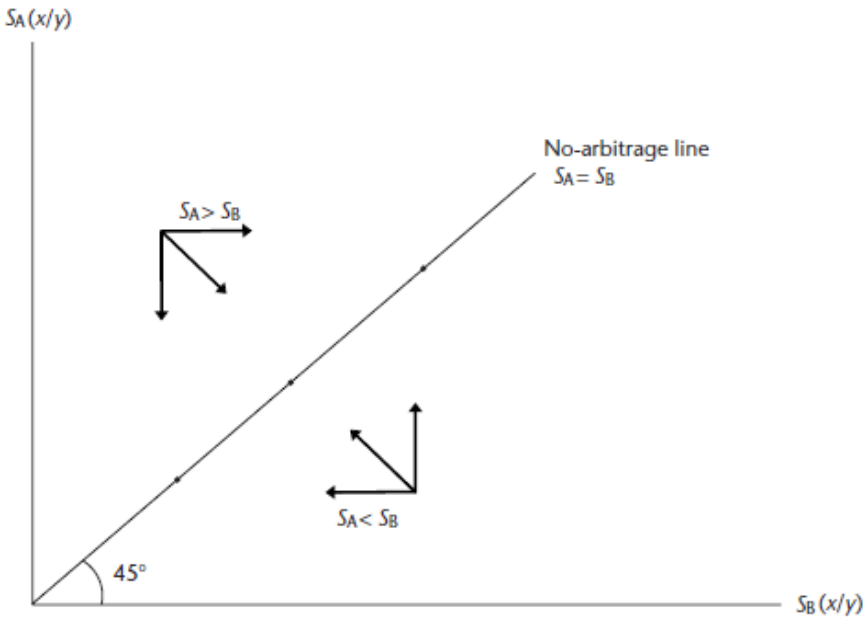

Figura 3.2: Linea di non-arbitraggio (arbitraggio bilaterale)

condizione dev'essere soddisfatta

$$S_A - S_B > (\beta_A + \beta_B) \tag{3.6}$$

il che sta ad indicare che la differenza dei prezzi dei due mercati dev'essere maggiore della somma delle commissioni sostenute per l'esecuzione delle transazioni. In questo modo, la condizione di non-arbitraggio con la presenza di commissioni di intermediazione fisse è data data

$$S_A = S_B + (\beta_A + \beta_B) \tag{3.7}$$

Analogamente, se l'arbitraggista vuole ottenere un profitto comprando y in A e vendendolo in B, la seguente condizione deve essere soddisfatta

$$S_B - S_A > (\beta_A + \beta_B) \tag{3.8}$$

e per quanto riguarda la condizione di non-arbitraggio, questa diventa

$$S_A = S_B - (\beta_A + \beta_B) \tag{3.9}$$

La Figura 3.3 mostra l'effetto dell'arbitraggio bilaterale in presenza di commissioni fisse quando un arbitraggista vuole comprare y in B e venderlo in A. La domanda in B e l'offerta in A aumentano, incrementando così il tasso di cambio in B e facendolo, invece, crollare in A. In questo caso, tuttavia, la finestra di arbitraggio non si chiude quando i tassi di cambio proposti da i due exchanges si equivalgono, bensì quando la differenza fra i due tassi di cambio è uguale alla somma delle commissioni di entrambi i mercati.

La Figura 3.4 mostra cosa succede alla linea di non-arbitraggio quando c'è la presenza di commissioni fisse. Una fascia, di ampiezza $2(\beta_A + \beta_B)$, verrà creata attorno alla linea

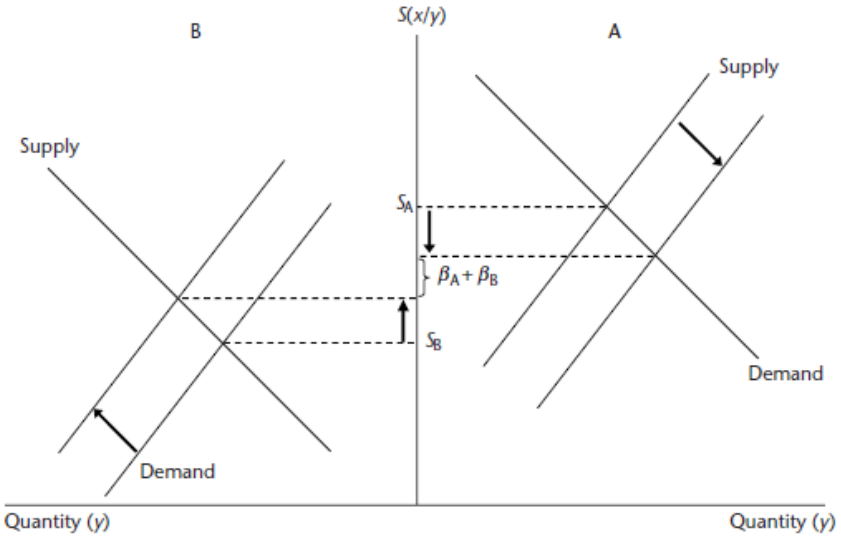

Figura 3.3: Effetto dell'arbitraggio bilaterale in presenza di commissioni

originale di non-arbitraggio. Il limite superiore della fascia è definito dall'equazione (3.8), mentre il limite inferiore è definito dall'equazione (3.9). I punti compresi all'interno di questa fascia ma che non ricadono esattamente sulla linea di non-arbitraggio originale, indicano che, seppur i tassi di cambio non siano uguali fra i due mercati, il profitto dall'operazione di arbitraggio non è comunque possibile in quanto questo verrebbe completamente assorbito dalle commissioni. I punti che ricadono invece al di fuori definiscono delle occasioni profittevoli di arbitraggio. Al di sopra del limite superiore, l'arbitraggio produrrà profitto comprando y in B e vendendolo in A; al di sotto del limite inferiore invece, l'arbitraggio sarà profittevole comprando y in A e vendendolo in B.

Si assuma ora che le commissioni di intermediazione dipendano dalla dimensione dell'operazione di compravendita, in modo tale che esse siano addebitate al tasso β_A e β_B rispettivamente nel mercato A e B. Se $S_A > S_B$ allora l'arbitraggista potrà comprare

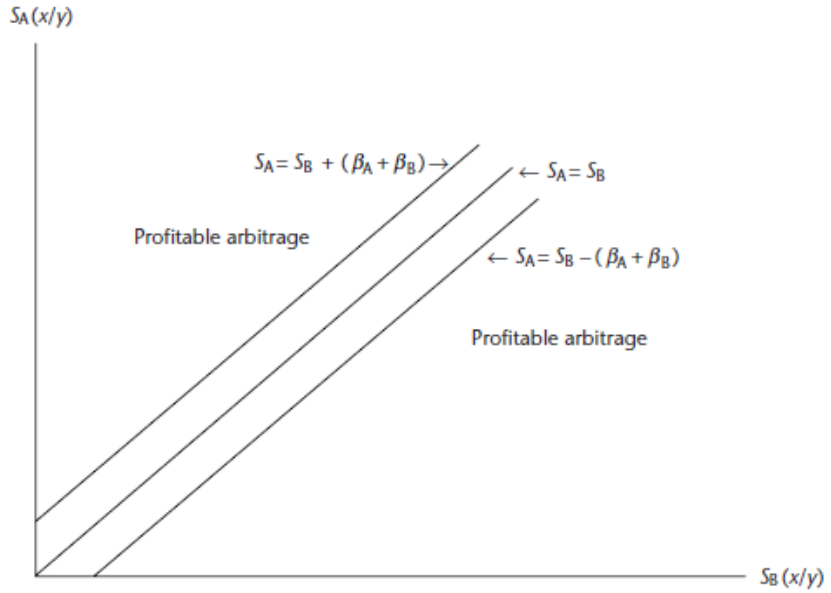

Figura 3.4: Linea di non-arbitraggio (arbitraggio bilaterale in presenza di commissioni)

y in B e rivenderlo in A. In questo caso il profitto realizzato con l'arbitraggio è

$$\pi = S_A(1 - \beta_A) - S_B(1 + \beta_B) \tag{3.10}$$

il che significa che l'arbitraggio sarà profittevole ($\pi > 0$) se

$$S_A > S_B \left[\frac{(1 + \beta_B)}{(1 - \beta_A)} \right] \tag{3.11}$$

In modo alternativo, se $S_A < S_B$, allora l'arbitraggista comprerà y in A e lo venderà in B; in questo caso il profitto risulta

$$\pi = S_B(1 - \beta_B) - S_A(1 + \beta_A) \tag{3.12}$$

che può essere rivista per evidenziare che l'arbitraggio sarà profittevole se

$$S_A < S_B \left[\frac{(1 - \beta_B)}{(1 + \beta_A)} \right] \tag{3.13}$$

In questo modo la linea di non-arbitraggio associata all'equazione (3.11) e (3.13) sarà rispettivamente

$$S_A = S_B \left[\frac{(1 + \beta_B)}{(1 - \beta_A)} \right] \tag{3.14}$$

e

$$S_A = S_B \left[\frac{(1 - \beta_B)}{(1 + \beta_A)} \right] \tag{3.15}$$

L'equazione (3.5) mostra cosa succede alla linea di non-arbitraggio in questo caso: si può notare che finché $0 < \beta_A < 1$ e $0 < \beta_B < 1$, è conseguentemente logico che $\frac{1+\beta_B}{1-\beta_A} > 1$ e $\frac{1-\beta_B}{1+\beta_A} < 1$. Ragionando in modo schematico, le equazioni (3.14) e (3.15) possono essere rappresentate, come nella Figura 3.5, da due linee che si intersecano sull'origine con la linea originale di non-arbitraggio, con una delle due più ripida e una più piatta. I punti che ricadono nell'area triangolare definita dalle tre rette indicano l'area di arbitraggio non profittevole, in quanto ogni profitto realizzato verrà assorbito totalmente dalle commissioni dei due mercati. Ogni punto al di sopra o al di sotto di tale area triangolare indica un'opportunità di arbitraggio profittevole.

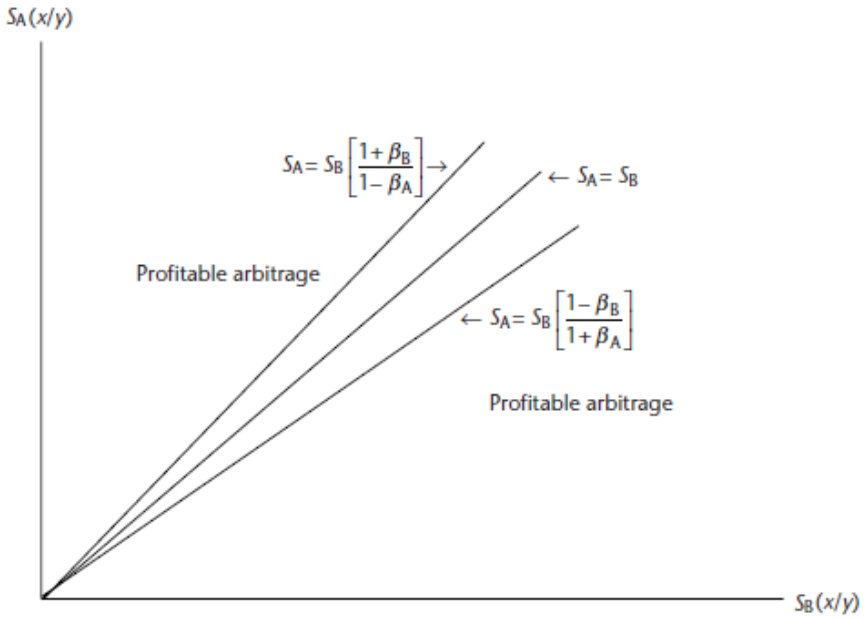

Figura 3.5: Linea di non-arbitraggio (arbitraggio bilaterale in presenza di commissioni progressive)

3.3.3 Arbitraggio bilaterale in presenza di tassazione

In questo paragrafo verranno considerate due tipologie di imposte: la tassa sul *capital gain* e la *Tobin tax*[1].

Si supponga che questa imposta sia calcolata in base ai profitti realizzati da un'operazione di arbitraggio bilaterale nel mercato finanziario dove il profitto viene ottenuto. E' facile così vedere come la presenza di una tassa sul *capital gain* non abbia alcun effetto sulla linea di non-arbitraggio, in quanto l'unico effetto dell'imposta è quello di ridurre il profitto ricevuto dall'arbitraggista. Finché la discrepanza fra i tassi di cambio permane, l'arbitraggio sarà profittevole, anche se minore rispetto ad uno scenario privo di tassazione. La finestra di arbitraggio non si chiuderà, quindi, finché tale discrepanza non scomparirà.

[1] Per quanto riguarda l'imposta sul *capital gain*, gli interessi degli investimenti sono soggetti alla tassazione sulle rendite finanziarie, in misura differente in base alla tipologia dell'investimento stesso: verrà così pagata una tassa più bassa per i rendimenti dei Titoli di Stato e dei buoni fruttiferi postali, più alta invece per i rendimenti dei conti deposito e correnti, delle azioni in Borsa, dei vari trading, del forex, dell'acquisto di metalli preziosi, delle obbligazioni bancarie, degli ETF, le assicurazioni sulla vita e le polizze d'investimento. Per quanto riguarda le cyrptovalute, attualmente non è presente una regolamentazione precisa e quindi pare si debba procedere per analogia, paragonandole alla seconda categoria di *capital gain*, tassata attualmente al 26%; la criptovaluta è definita valuta virtuale, sia che si tratti di bitcoin che di altre valute. Le operazioni in valuta virtuale sono esenti da Iva. Per le persone fisiche che detengono le valute virtuali al di fuori dell'attività di impresa l'agenzia delle Entrate ha dichiarato che le operazioni di acquisto e vendita di valuta non generano redditi, mancando la finalità speculativa. In dottrina tuttavia alcuni ritengono che sia la nuova definizione normativa di valuta virtuale che l'evoluzione del mercato che ha portato ad un investimento di massa su tali criptovalute possano assimilare tali operazioni ad investimenti finanziari e che quindi debba essere applicata la tassazione sui capital gain ovvero a certe condizioni quella sulla speculazione sulle valute [32].
La cosiddetta *Tobin Tax*, invece, si applica ai trasferimenti di proprietà di azioni e strumenti partecipativi emessi da società residenti nel territorio dello Stato. Nello specifico saranno soggette alla tassa tutte le transazioni (a carico del solo acquirente) su azioni di Società italiane quotate aventi capitalizzazione superiore a 500 milioni di euro (al 30 Novembre di ogni anno), a prescindere dal paese dal quale proviene l'ordine o del mercato in cui tali società sono quotate. Sono soggette a *Tobin Tax* tutte le società non quotate sui mercati regolamentati anche se sotto capitalizzate. Può risultare interessante analizzare anche questa tipologia d'imposte in quanto, nel mondo della Blockchain, le criptovalute possono considerarsi anche come strumenti partecipativi dei progetti che le hanno emesse [33].
Per quanto riguarda l'analisi dal punto di vista dell'arbitraggio, verrà prima di tutto analizzato l'andamento del profitto in presenza di tassazione per *capital gain*.

Definendo l'ammontare dell'imposta sul *capital gain* come τ, allora il profitto ante oneri finanziari ottenuto comprando y in B e vendendolo in A è

$$\pi = (1 - \tau)[S_A - S_B] \tag{3.16}$$

e in questo caso, la linea di non-arbitraggio sarà data dall'equazione

$$(1 - \tau)[S_A - S_B] = 0 \tag{3.17}$$

la quale equivale all'Equazione (3.14). Si può notare inoltre che $\delta\pi/\delta\tau = 0$. Per quanto riguarda la Tobin tax invece, era stata introdotta come una misura per ridurre la volatilità nei mercati finanziari esteri; l'imposta è calcolata come percentuale sul valore della transazione, e quindi avrà lo stesso effetto delle commissioni di intermediazione, viste nel paragrafo precedente.

3.3.4 Arbitraggio bilaterale in presenza di controllo sui movimenti dei capitali (capital controls)

Quando vi è l'imposizione di un qualche tipo di controllo sui movimenti dei capitali da parte di uno dei mercati finanziari presi in considerazione, l'arbitraggio non sarà possibile, e le discrepanze nei tassi di cambio rimarranno tali. Nulla permetterà quindi lo spostamento delle curve di domanda e offerta come in Figura 3.1. Tuttavia, se il controllo sui movimenti di capitale è parziale, la quantità di capitale di cui è consentito il trasferimento da un mercato ad un altro potrebbe non essere sufficiente per spostare la curva della domanda e dell'offerta in modo tale da eliminare la discrepanza fra i tassi di cambio delle valute. In questo caso, i tassi di cambio nei due mercati finanziari si avvicineranno, ma non saranno mai uguali. Questa situazione è spiegata in Figura 3.6.

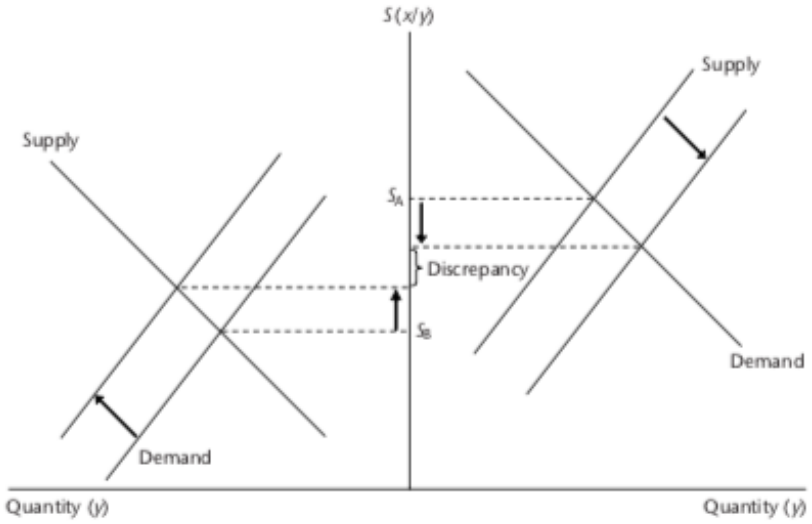

Figura 3.6: Linea di non-arbitraggio (arbitraggio bilaterale in presenza di controllo sui movimenti dei capitali)

3.3.5 Arbitraggio bilaterale in presenza del bid-ask spread

Si considerino $S_{b,A}$ e $S_{b,B}$ come i prezzi *bid* (in italiano, prezzo "denaro"), e $S_{a,A}$ e $S_{a,B}$ come i prezzi *ask* (prezzo "lettera"), quotati rispettivamente nei mercati A e B. In presenza di uno spread fra *bid* e *ask*, gli arbitraggisti possono comprare (da i *market makers*) ad un tasso *ask* più alto per poi vendere (sempre a dei *market makers*) ad un tasso *bid* più basso (si ricorda che $S_b < S_a$). Pertanto, il tasso *bid* verrà determinato dalla domanda dei *market-makers* e dall'offerta degli arbitraggisti. Al contrario, il tasso *ask* è determinato dalla domanda degli arbitraggisti e dall'offerta dei *market-makers*. I tassi *bid* e *ask* possono essere messi in relazione con l'equazione

$$S_a = S_b(1 + m) \tag{3.18}$$

dove m è lo spread fra il *bid* e l'*ask* espresso in termini percentuali rispetto al tasso *bid*. Per semplicità, assumeremo che lo spread *bid-ask* sia lo stesso per entrambi i mercati finanziari. Si supponga quindi che un arbitraggista voglia comprare y in B per poi

venderlo in A. In questo caso, l'arbitraggio sarà profittevole se

$$\pi = S_{b,A} - S_{a,B} > 0 \tag{3.19}$$

il che significa che la condizione di non-arbitraggio è data da

$$S_{b,A} = S_{a,B} \tag{3.20}$$

Questa situazione viene mostrata nella Figura 3.7; gli arbitraggisti comprando y in B al $S_{a,B}$ e lo rivendono in A al $S_{b,A}$. Tale processo comporta uno spostamento nella curva di domanda degli arbitraggisti in B, causando a sua volta un aumento di $S_{a,B}$ e allo spostamento della curva d'offerta degli arbitraggisti in A, causando quindi una diminuzione di $S_{b,A}$. Tale processo continua finché i due tassi di cambio sono uguali; se solamente questi cambiamenti vengono attuati, lo spread *bid-ask* aumenterà in entrambi i mercati finanziari, anche se non c'è nessuna ragione per cui ciò debba accadere. Per mantenere lo spread *bid-ask* allo stesso livello, $S_{b,B}$ deve aumentare e $S_{a,A}$ deve diminuire. Il motivo per cui i market makers trovano profittevole aumentare l'offerta di y, proporzionalmente alla crescita di $S_{a,B}$ verrà spiegato nelle prossime righe. Per far ciò, devono ottenere grandi quantità di y comprandolo dagli offerenti. Così, la curva di domanda dei market makers si sposta, portando quindi ad un incremento di $S_{b,B}$. In modo simile, ad una diminuzione di $S_{b,A}$, i market makers troveranno più conveniente comprare y, e la curva di offerta si sposterà verso sinistra, provocando una diminuzione di $S_{a,A}$. Si consideri ora la condizione di non-arbitraggio in presenza dello spread *bid-ask*. La figura 3.8 mostra un piano cartesiano completo, nel quale il quadrante 1 e 3 mostrano la condizione di non-arbitraggio, mentre i quadranti 2 e 4 mostrano la relazione tra i prezzi *bid* e *ask*.

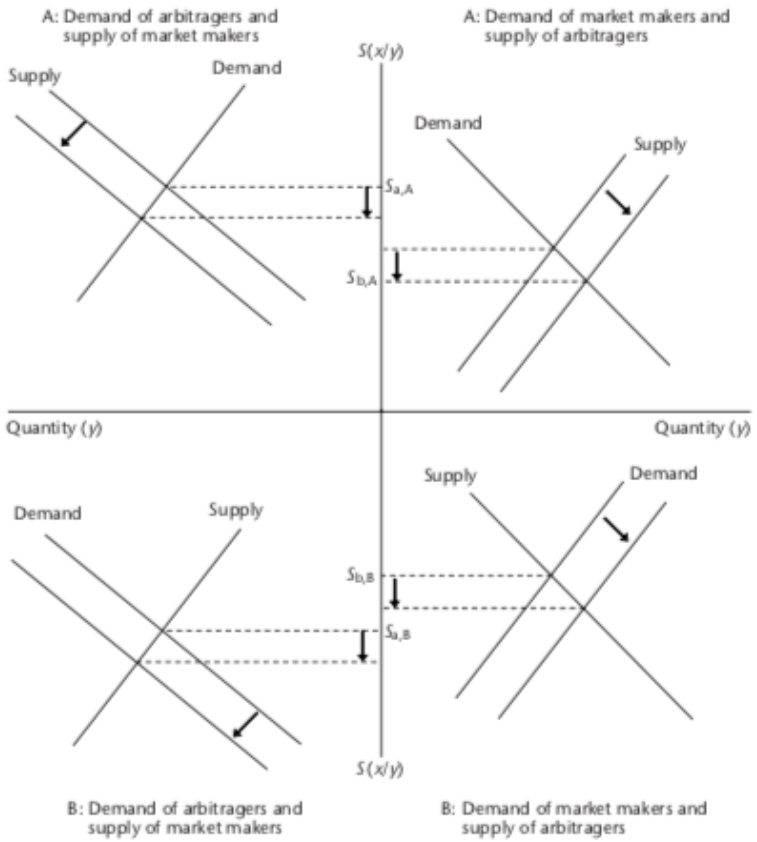

Figura 3.7: Effetto dell'arbitraggio bilaterale in presenza di uno spread *bid-ask*

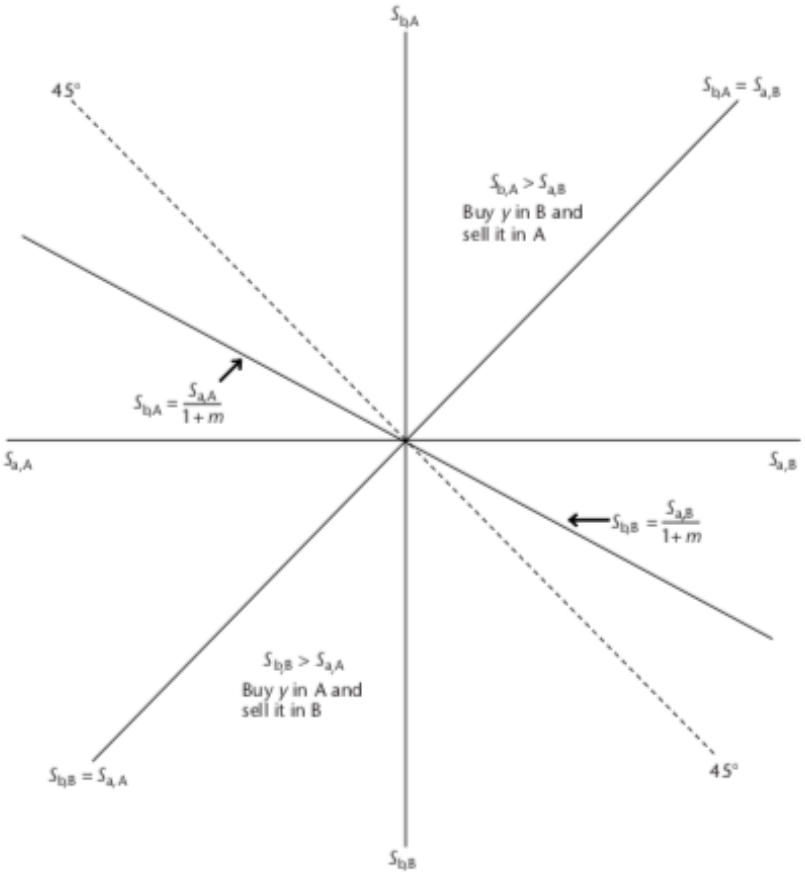

Figura 3.8: Condizione di non-arbitraggio in presenza di un bid-ask spread

Si può notare come la linea che rappresenta questa relazione sia meno ripida di una retta a 45°, questo perché il prezzo *bid* è sempre più basso del prezzo *ask*. Il primo quadrante mostra la linea di non-arbitraggio rappresentata dall'Equazione (3.20). Ogni punto sopra la linea implica un arbitraggio profittevole, con il profitto dato dall'Equazione 3.19. Nel terzo quadrante, i punti sotto la linea di non-arbitraggio rappresentano un'opportunità di arbitraggio profittevole, comprando y da A per poi venderlo in B. In questo caso il profitto sarà;

$$\pi = S_{b,B} - S_{a,A} \tag{3.21}$$

L'effetto dello spread *bid-ask* è quello di ridurre la profittabilità dell'arbitraggio, in quanto lo spread rimane un costo di transazione. Tornando all'Equazione (3.3), che definisce il profitto dell'arbitraggio come la differenza fra il tasso di cambio in A (il prezzo di vendita) e il tasso nel mercato B (il prezzo d'acquisto). Questi prezzi, in quell'equazione, non erano definiti come prezzi *bid* o *ask*, quindi si supponga che siano prezzi medi, il che significa:

$$S_A = \frac{1}{2}[S_{b,A} + S_{a,A}] \tag{3.22}$$

$$S_B = \frac{1}{2}[S_{b,B} + S_{a,B}] \tag{3.23}$$

Il profitto dell'arbitraggio in assenza ed in presenza dello spread *bid-ask* è dato rispettivamente dalle Equazioni (3.3) e (3.19). Finché per definizione:

$$S_{b,A} < S_A \tag{3.24}$$

e

$$S_{a,B} > S_B \tag{3.25}$$

seguirà che

$$S_{b,A} - S_{a,B} < S_A - S_B \tag{3.26}$$

il che significa che la presenza di uno spread *bid-ask* riduce la profittabilità dell'arbitraggio, in quanto l'arbitraggista dovrà pagare ad un prezzo più alto e rivendere ad un prezzo più basso invece che viceversa.

3.3.6 Arbitraggio bilaterale considerando tutte le condizioni

Si consideri la profittabilità dell'arbitraggio bilaterale in presenza dello spread *bid-ask*, di commissioni fisse di intermediazione e della tassa sul *capital gain*; si prenda inoltre la situazione nella quale l'arbitraggista compra y in B e lo rivende in A (equazioni (3.19) e (3.20). In presenza di una commissioni fissa, β_A e β_B, e una tassa sul *capital gain*, τ, la quale si assume sia uguale in entrambi i mercati finanziari, il profitto dell'arbitraggio si riduce a

$$\pi = S_{b,A}(1-\tau) - \beta_A - S_{a,B}(1+\tau) - \beta_B = [S_{b,A} - S_{a,B}](1-\tau) - (\beta_A + \beta_B) \tag{3.27}$$

che significa una riduzione ulteriore del profitto, in quanto interviene anche una riduzione dello spread *bid-ask*. L'Equazione (3.27) indica una condizione di non arbitraggio espressa come

$$S_{b,A} - S_{a,B} = \frac{\beta_A + \beta_B}{1-\tau} \tag{3.28}$$

da cui si può dire che, per un arbitraggio bilaterale profittevole, il gap tra il prezzo *bid* in A e il prezzo *ask* in B dev'essere maggiore di $(\beta_A + \beta_B)/(1-\tau)$.

3.4 Arbitraggio triangolare

L'arbitraggio triangolare, conosciuto anche come arbitraggio trilaterale o a tre vertici, agisce nel seguente modo. Date tre valute (x, y e z), esistono tre possibili tassi di cambio: $S(x/y)$, $S(x/z)$ e $S(y/z)$. In quanto si sta contrattando in tre tassi di cambio differenti, verrà utilizzata la notazione originale, la quale mostra l'unità di misura del prezzo *spot*. Possiamo dire che i tre tassi di cambio sono in equilibrio se esiste la seguente condizione

[34]

$$S(x/y) = \frac{S(x/z)}{S(y/z)} \tag{3.29}$$

Si osservi ora cosa accade se un arbitraggista prova a ottenere profitto muovendosi da una valuta ad un altra, fino a ritornare alla prima valuta utilizzata. Se l'arbitraggista conclude l'operazione con un'unità della moneta con cui era partito, allora l'arbitraggio sarà profittevole. In generale, se la Condizione (3.29) viene violata allora sarà possibile effettuare profitto muovendosi in una determinata direzione e una perdita muovendosi in quella opposta. Quindi, partendo con un'unità della valuta x, si può eseguire un'operazione di arbitraggio:

1. Vendendo x e comprando y, ottenendo $\frac{1}{S(x/y)}$ unità di y;
2. Vendendo y e comprando z, ottenendo $\frac{1}{S(x/y)S(y/z)}$ unità di z;
3. Vendendo z e comprando x, ottenendo $\frac{S(x/z)}{S(x/y)S(y/z)}$ unità di x.

Il profitto realizzato da quest'operazione (misurato in unità di x) è dato da

$$\pi = \left(\frac{S(x/z)}{S(x/y)S(y/z)}\right) - 1 \tag{3.30}$$

Se la condizione rappresentata dalla formula (3.29) è valida, allora il profitto sarà $\pi = 0$; questa formula diventa quindi la condizione di non-arbitraggio. Tuttavia, se

$$S(x/y) < \frac{S(x/z)}{S(y/z)} \tag{3.31}$$

seguirà che

$$S(x/z) > S(x/y)S(y/z) \tag{3.32}$$

ciò significa che $\pi > 0$. Così, se la condizione di non-arbitraggio (3.29) viene violata, in modo tale che (3.31) sia valida, allora l'arbitraggio triangolare sarà profittevole seguendo la sequenza: $x \to y \to z \to x$. Vediamo quindi cosa succede se un arbitraggista segue la sequenza $x \to y \to z \to x$, partendo sempre con un'unità di x. L'operazione consisterà in

1. Vendendo x e comprando z, ottenendo $1/S(x/z)$ unità di z;
2. Vendendo y e comprando y, ottenendo $1/S(y/z)S(x/z)$ unità di z;
3. Vendendo y e comprando x, ottenendo $\frac{S(y/z)}{S(x/y)S(x/z)}$ unità di x.

Il profitto realizzato da questa operazione sarà

$$\pi = \left(\frac{S(y/z)S(x/y)}{S(x/z)} \right) - 1 \tag{3.33}$$

E' quindi ovvio che se la relazione (3.29) è valida, allora $\pi = 0$. In questo caso, un arbitraggio profittevole viene individuato tramite una violazione della condizione (3.29) in modo tale che

$$S(x/y) > \frac{S(x/z)}{S(y/z)} \tag{3.34}$$

che insieme all'equazione (3.34) implica che

$$S(y/z)S(x/y) > S(x/z) \tag{3.35}$$

che sta a significare $\pi > 0$.

Come anche nell'arbitraggio bilaterale, quello triangolare porta al ripristino della condizione di non-arbitraggio attraverso i cambiamenti nella domanda e nell'offerta per le tre valute. Vediamo ora cosa succede nel primo caso, rappresentato dalla relazione

(3.31). Con l'aiuto della Figura 3.9, possiamo osservare che ognuna delle tre fasi provoca un cambiamento nelle forze della domanda e dell'offerta, in questo modo

1. Un aumento nella domanda per y (l'offerta di x), provoca l'incremento di $S(x/y)$;

2. Un aumento nella domanda per z (l'offerta di y), provoca l'incremento di $S(y/z)$;

3. Un aumento nella domanda per x (l'offerta di z), provoca la diminuzione di $S(x/z)$.

Questi cambiamenti permetteranno alla condizione di equilibrio di ristabilirsi.

3.4.1 Arbitraggio triangolare in presenza di uno spread bid-ask

Vediamo cosa succede invece se si vuole seguire la sequenza $x \to z \to y \to x$ in presenza di uno spread *bid-ask*. L'operazione consiste nei seguenti steps:

1. Vendere x per comprare z al prezzo $S_a(x/z)$ per ottenere $1/[S_a(x/z)]$ unità di z;

2. Vendere z per comprare y al prezzo $S_b(y/z)$ per ottenere $S_b(y/z)/[S_a(x/z)]$ unità di y;

3. Vendere y per comprare x al prezzo $S_a(x/y)$ per ottenere $S_a(x/y)S_b(y/z)/[S_a(x/z)]$ unità di x.

Per rendere profittevole quest'operazione, devono essere soddisfatta le seguenti condizioni

$$\frac{S_a(x/y)S_b(y/z)}{S_a(x/z)} > 1 \tag{3.36}$$

da cui consegue

$$\pi = \frac{S_a(x/y)S_b(y/z)}{S_a(x/z)} - 1 \tag{3.37}$$

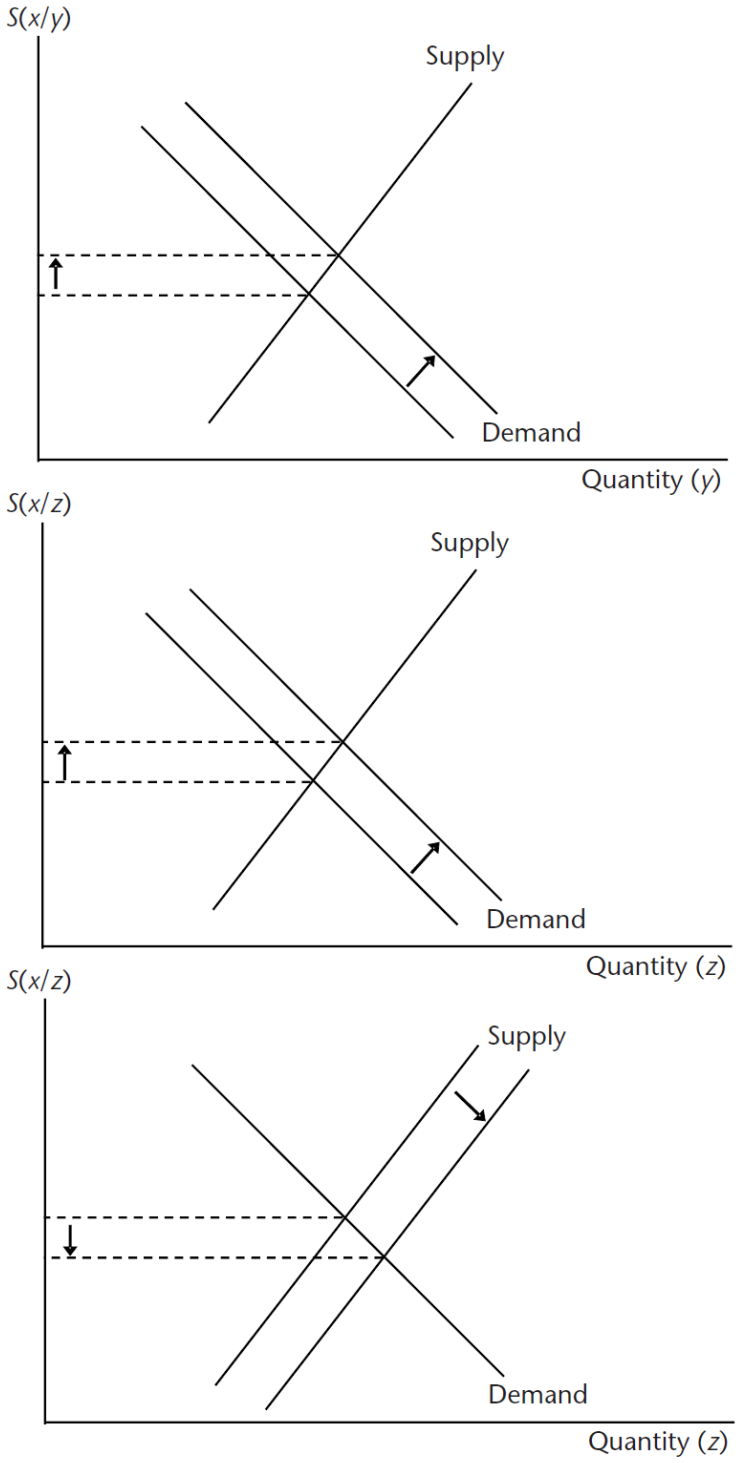

Figura 3.9: Effetto dell'arbitraggio triangolare sulla domanda e sull'offerta

in questo caso, la condizione di non-arbitraggio sarà

$$S_a(x/y) = \frac{S_a(x/z)}{S_b(y/z)} \tag{3.38}$$

Analogamente, può essere osservato che la sequenza $x \rightarrow y \rightarrow z \rightarrow x$ può essere profittevole se

$$\frac{S_b(x/z)}{S_b(x/y)S_a(y/z)} > 1 \tag{3.39}$$

da cui consegue

$$\pi = \frac{S_b(x/z)}{S_b(x/y)]S_a(y/z)} - 1 \tag{3.40}$$

in questo caso invece, la condizione di non-arbitraggio sarà

$$S_b(x/y) = \frac{S_b(x/z)}{S_a(y/z)} \tag{3.41}$$

Le equazioni (3.38) e (3.41) vengono usate per calcolare i cosiddetti *cross exchange rates* sia per il *bid* che per l'*ask* quando la valuta z è il numeratore.

3.5 Arbitraggio multivalutario

Si consideri ora un'operazione di arbitraggio che include quattro valute: x_1, x_2, x_3 e x_4, la quale segue la sequenza $x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \rightarrow x_1$. L'arbitraggio consiste nelle seguenti fasi:

1. Comprare x_2 vendendo x_1 al prezzo $S(x_1/x_2)$ per ottenere $1/S(x_1/x_2)$ unità di x_2;

2. Comprare x_3 vendendo x_2 al prezzo $S(x_2/x_3)$ per ottenere $1/S(x_1/x_2)S(x_2/x_3)$ unità di x_3;

3. Comprare x_4 vendendo x_3 al prezzo $S(x_3/x_4)$ per ottenere $1/S(x_1/x_2)\, S(x_2/x_3)S(x_3/x_4)$ unità di x_4;

4. Comprare x_1 vendendo x_4 al prezzo $S(x_1/x_4)$ per ottenere $S(x_1/x_4)/S(x_1/x_2)S(x_2/x_3)$ $S(x_3/x_4)$ unità di x_1.

Come viene illustrato in [34], questa operazione sarà profittevole se

$$\pi = \frac{S(x_1/x_4)}{S(x_1/x_2)S(x_2/x_3)S(x_3/x_4)} - 1 > 0 \tag{3.42}$$

in questo caso particolare, la condizione di non-arbitraggio sarà

$$S(x_1/x_4) = S(x_1/x_2)S(x_2/x_3)S(x_3/x_4) \tag{3.43}$$

oppure

$$S(x_4/x_1)S(x_1/x_2)S(x_2/x_3)S(x_3/x_4) = 1 \tag{3.44}$$

In generale, ogni arbitraggio che include n-valute è profittevole se la seguente condizione di non-arbitraggio viene violata

$$S(x_1/x_4)S(x_1/x_2)S(x_2/x_3)S(x_3/x_4) \cdots S(x_n - 1/x_n)S(x_n/x_1) = 1 \tag{3.45}$$

tramite l'Equazione (3.45) è possibile rappresentare ogni tipo di arbitraggio, anche quello bilaterale; infatti, se $n = 2$, la condizione di non-arbitraggio si riduce a

$$S(x_1/x_2)S(x_2/x_1) = 1 \tag{3.46}$$

Come riporta Chacholiades nel paper "*The sufficiency of three-point arbitrage to ensure consistent cross rates of exchange*" del 1971 [35], se un arbitraggio triangolare non risulta

profittevole, allora non lo sarà neanche con n-valute. Questo significa che per mantenere la condizione di non-arbitraggio definita da (3.45), è sufficiente che la condizione sia

$$S(x_1/x_2)S(x_2/x_3)S(x_3/x_1) = 1 \tag{3.47}$$

La conferma di questa relazione è di natura matematica: se l'arbitraggio con $(n-1)$-valute non genera profitto, allora non lo sarà neanche l'arbitraggio con n-valute. Se quindi, per assunzione, l'arbitraggio con $(n-1)$ valute non è profittevole, allora la seguente relazione verrà mantenuta tale

$$S(x_1x_2)S(x_2x_3)S(x_3x_4)\cdots S(x_{n-\star}x_{n-1})S(x_{n-1}x_1) = 1 \tag{3.48}$$

Dividendo l'equazione (3.44) per la (3.48), si otterrà

$$S(x_{n-1}/x_n)S(x_n/x_1)S(x_{n-1}/x_1) = 1 \tag{3.49}$$

tale relazione, con $n = 3$, sarà equivalente all'equazione (3.47) in quanto $S(x_1/x_2) = 1/S(x_2/x_1)$. Così, se le Equazioni (3.47) e (3.48) vengono rispettate, anche la (3.45) dovrà essere soddisfatta, il che prova la relazione matematica che lega queste tre equazioni.

3.5.1 Arbitraggio multivalutario in presenza dello spread bid-ask

In presenza di uno spread *bid-ask*, l'operazione di arbitraggio si articola nelle seguenti fasi

1. Comprare x_2 vendendo x_1 al prezzo $S_a(x_1/x_2)$ per ottenere $1/S_a(x_1/x_2)$ unità di x_2;

2. Comprare x_3 vendendo x_2 al prezzo $S_a(x_2/x_3)$ per ottenere $1//S_a(x_1/x_2)S_a(x_2/x_3)$ unità di x_3;

3. Comprare x_4 vendendo x_3 al prezzo $S_a(x_3/x_4)$ per ottenere $1/S_a(x_1/x_2)S_a(x_2/x_3)S_a(x_3/x_4)$ unità di x_4;

4. Comprare x_1 vendendo x_4 al prezzo $S_b(x_1/x_4)$ per ottenere $\frac{S_b(x_1/x_4)}{S(x_1/x_2)S(x_2/x_3)}\ S(x_3/x_4)$ unità di x_1.

Questa operazione risulterà profittevole se

$$\pi = \frac{S_b(x_1/x_4)}{S(x_1/x_2)S(x_2/x_3)S(x_3/x_4)} - 1 > 0 \tag{3.50}$$

come abbiamo visto anche per le altre situazioni di arbitraggio, la condizione di non-arbitraggio sarà

$$S_b(x_1/x_4) = S(x_1/x_2)S(x_2/x_3)S(x_3/x_4) \tag{3.51}$$

o anche

$$S_a(x_1/x_2)S_a(x_2/x_3)S_a(x_3/x_4)S_a(x_4/x_1) = 1 \tag{3.52}$$

tenendo sempre presente che $S_b(x_1/x_4) = 1/S_a(x_4/x_1)$, il che permette di convertire l'equazione (3.51) in (3.52). Se lo spread *bid-ask* (indicato con m ed espresso in percentuale rispetto al prezzo *bid*) è il medesimo per tutti i tassi di cambio, la condizione di non-arbitraggio diventa

$$S_b(x_1/x_4)S_b(x_2/x_3)S_b(x_3/x_4)S_b(x_1/x_4)[(1+m)^4] = 1 \tag{3.53}$$

da cui si può dedurre che la condizione di non-arbitraggio con n-valute ed in presenza di uno spread *bid-ask* è data dall'equazione

$$S_a(x_1/x_2)S_a(x_2/x_3)S_a(x_3/x_4)\cdots S_a(x_n/x_1) = 1 \tag{3.54}$$

che può essere espressa anche in funzione del prezzo *bid*

$$S_b(x_1/x_2)S_b(x_2/x_3)S_b(x_3/x_4)\cdots S_b(x_n/x_1)[(1+m)^n] = 1 \tag{3.55}$$

3.6 Rischi pratici dell'arbitraggio

Nel mondo accademico, l'arbitraggio è spesso definito come un operazione priva di rischi. Tuttavia, nel mondo reale, non esiste una situazione priva di ogni rischio, e l'obiettivo di questo capitolo è spiegarne il motivo. Il primo rischio che verrà analizzato è il *rischio di liquidità*, dovuto al fatto che alcune strategie di arbitraggio richiedono di prendere una posizione opposta rispetto ai movimenti del mercato; qui l'arbitraggista può trovarsi nella situazione di non avere abbastanza liquidità e questo può forzarlo a chiudere le posizioni e non ottenere alcun profitto. Il secondo rischio sarà il *rischio di controparte*, in quanto l'attuale sistema finanziario richiede l'utilizzo di banche, brokers e altre istituzioni. Il rischio del fallimento di uno di questi istituti può causare potenziali perdite. Il terzo rischio analizzato sarà il *rischio di esecuzione*, il quale è proporzionale allo sviluppo della tecnologia utilizzata. Ci potrebbero essere quindi situazioni di ritardo o addirittura interruzione degli ordini i quali portano a situazioni scomode, spesso risultando in una perdita. Il quarto ed ultimo rischio è il cosiddetto *slippage*, cioè il rischio dato dalla probabilità che il prezzo a cui un'operazione viene eseguita sia diverso dal prezzo atteso dall'investitore.

3.6.1 Rischio di liquidità

Al contrario della supposizione che per l'arbitraggio non servano molti capitali, la maggior parte delle occasioni reali di arbitraggio richiedono un'elevata quantità di capitale, soprattutto se si introduce l'utilizzo di strumenti derivati, che potrebbero portare ad una *margin call* o alla necessità di coprire la posizione con ulteriore capitale in base

ai movimenti del mercato [36].
In altre parole, se il movimento di mercato è d'intensità alta, l'arbitraggista potrebbe essere obbligato ad aggiungere ulteriore capitale nella posizione, e nel caso in cui non avesse più capitali a disposizione, la posizione si chiuderebbe provocando una perdita [31].
Quando l'arbitraggista è un fondo (o un istituzione) e utilizza i fondi messi a disposizione dai suoi investitori per eseguire l'operazione di arbitraggio, il cosiddetto valore *mark-to-market*[2] del fondo potrebbe avere un'alta volatilità anche se l'operazione, fra tutte quelle comprese nel fondo, ha un basso profilo di rischio. Questi investitori potrebbero ritirare il loro investimento dal fondo, spinti dalla paura di una perdita, e ciò potrebbe non permettere più al fondo di garantire il margine minimo per mantenere una posizione, portando poi alla sua chiusura forzata e provocando una perdita [36].
Per la ragione menzionata nel precedente Paragrafo (3.6), i mercati molto volatili potrebbero non essere molto attrattivi per gli arbitraggisti in quanto, anche se spesso le operazioni di arbitraggio risultano più profittevoli, il rischio implicito di liquidità potrebbe scoraggiarli ad effettuare l'operazione. Alcune opportunità di arbitraggio possono infatti richiedere capitali significativi, senza menzionare il conseguente mantenimento del margine di sicurezza per le operazioni che includono anche l'utilizzo di strumenti derivati.

3.6.2 Rischio di controparte

Gli arbitraggisti sono obbligati ad aderire all'attuale sistema finanziario per condurre un'operazione di arbitraggio; devono avvalersi della collaborazone di intermediari, brokers, banche, piattaforme e altre istituzione per eseguire i trades e depositare il loro

[2] *Mark-to-market* significa "valutare secondo il mercato". Le attività finanziarie (ma anche quelle reali) possono essere valutate secondo il costo storico (o costo di acquisizione), secondo una procedura detta di "costo corrente", che sarebbe il costo storico riportato ad oggi mediante un indice dei prezzi, o secondo il prezzo di mercato [37]

capitale. Affidare i propri fondi ad una delle istituzioni precedentemente menzionate comporta un rischio di controparte. Tale rischio, in altre parole, è la probabilità che questi istituti vadano in default; e per questa ragione, la maggioranza delle opportunità di arbitraggio non sono completamente prive di rischio, difatti ci sarà sempre il rischio del fallimento di un exchange, di una banca, o di un soggetto coinvolto nell'esecuzione dell'operazione di arbitraggio.

3.6.3 Rischio di esecuzione

I rischi di esecuzione sono rischi legati allo stato della tecnologia contemporanea, alle loro limitazioni, imperfezioni e caratteristiche. Questi rischi possono trasformare le operazioni di arbitraggio in operazioni rischiose, come, ad esempio, nel caso in cui vi sia un sufficiente ritardo fra la pubblicazione e l'effettiva esecuzione di un ordine a mercato. L'attuale sistema di trading elettronico è basato su un network centralizzato dove ogni elemento della rete introduce un potenziale ritardo addizionale. Questi ritardi spesso accadono in successione, formando un effetto a cascata; è quindi molto importante avere una conoscenza approfondita delle tecnologie utilizzate durante le operazioni di arbitraggio per poter ottimizzare la velocità del flusso di informazioni e minimizzare i ritardi nelle esecuzioni.

Velocità dei processori

Il processore, detto anche CPU (Central Processing Unit), è un sofisticato circuito elettronico in grado di eseguire un numero finito di operazioni per unità di tempo. Le CPUs possono contenere processori logici multipli, i quali possono eseguire istruzioni in parallelo, cioè nello stesso momento. L'algoritmo di arbitraggio per funzionare correttamente dovrebbe essere ottimizzato sulla velocità, in modo tale da limitare il rischio causato dai ritardi dei processi di calcolo.

Sfortunatamente, non è sempre possibile parallelizzare le operazioni di arbitraggio, in quanto gli ordini vengono processati su server remoti diversi. Per questa ragione, l'algoritmo su cui si basa l'arbitraggio dovrebbe essere ottimizzato sulla velocità, in modo tale da limitare il rischio causato dai ritardi dei processi di calcolo.

Latenza della rete

La latenza della rete è il tempo necessario ad una richiesta effettuata da un computer ad essere trasferita attraverso tutta le rete (che consiste in schede di rete, routers, bridges, hubs e altri strumenti di rete) per raggiungere la sua destinazione. Il protocollo più utilizzato al giorno d'oggi è il protocollo TCP/IP, e riesce a garantire l'integrità dei dati, il corretto ordine dei pacchetti di dati e a fornire la conferma di questi dati (l'indirizzo ricevente conferma di aver ricevuto i dati del mittente). Tuttavia, questo protocollo non garantisce un tempo fisso per il trasferimento dei dati; il tempo di trasferimento è variabile, intensificando la componente di rischio legata alla latenza della rete. Per questa ragione, è importante ottimizzare i moduli di comunicazione con la rete, in modo da permettere un trasferimento di dati veloce e senza interruzioni.

Malfunzionamenti hardware o software

Nel momento in cui vengono individuate le opportunità di arbitraggio, potrebbe avvenire un malfunzionamento a livello del software o anche nell'hardware. In quest'ultimo caso, i motivi potrebbero essere molteplici, fra cui un errore logico della CPU o anche il deterioramento di alcuni dati nella memoria fisica. Tali situazioni potrebbero essere causate dal problemi nel design della macchina o da eventi esterni, come la collisione di alcune particelle ad alta velocità con gli atomi presenti nella memoria della macchina, cambiando quindi la carica di un bit e corrompendo così i dati presenti. Guardando al trasferimento dei dati invece, un pacchetto potrebbe essere perso durante il suo trasferimento, o i dati al suo interno potrebbero essere corrotti. Per finire, c'è da

tenere conto anche di eventuali interruzioni della corrente che porterebbero al blackout dell'intero sistema, interrompendo ogni trasferimento d'informazione e potenzialmente causando una perdita. Nel caso del malfunzionamento del lato software, i problemi potrebbero ricadere sui driver della macchina, sul kernel del sistema operativo o su altre parti di codice che porterebbero a un crash dell'intero sistema, richiedendo così il riavvio forzato della macchina. Inoltre, anche l'algoritmo di arbitraggio potrebbe contenere degli errori nel suo design, sia dal punto di vista dell'algoritmo in sé che alla sua implementazione (comunicazione con le piattaforme di trading, problemi di sincronizzazione, fat finger errors, etc.). E' utile infine menzionare che ognuno di questi malfunzionamenti potrebbe accadere nel mezzo dell'esecuzione di un'operazione di arbitraggio, ad esempio fra l'esecuzione del primo e secondo ordine di un arbitraggio bilaterale. Questo porterrebbe l'arbitraggista a trovarsi una posizione aperta per un determinato strumento, con il forte rischio di vedersi chiudere la finestra di arbitraggio ed ottenere conseguentemente una perdita.

3.6.4 Slippage

Lo slippage, nell'arbitraggio, si riferisce alla differenza fra il prezzo atteso di un operazione e il prezzo al quale essa viene eseguita.
Lo slippage può accadere per due ragioni:

1. Dovuto ad un alta volatilità del mercato; quando i prezzi variano in rapida successione, c'è la probabilità che il prezzo mostrato nel momento in cui l'ordine viene inserito potrebbe non essere il prezzo viene eseguito effettivamente.

2. Risultante dal fatto che sul mercato è presente un ordine di eccessive dimensioni o il mercato ha scarsa liquidità per quello strumento.

In entrambi i casi, l'ordine potrebbe venir eseguito con diversi ordini presenti nell'order book, risultando nella media del prezzo di esecuzione, il quale è meno favorevole del

prezzo in cima al book. Lo slippage non si riferisce ad un determinato movimento positivo o negativo, ma può essere associato ad ogni differenza fra il prezzo atteso e il prezzo effettivo. Quando gli ordini vengono eseguiti, lo strumento viene acquistato o venduto al miglior prezzo disponibile.
Questo può essere più favorevole, uguale o svantaggioso rispetto alle aspettative iniziali; risultano quindi tre situazioni possibili: uno slippage positivo, assenza di slippage e uno slippage negativo. Lo slippage è un termine usato sia nel forex che nello stock market, e anche se mantentiene lo stesso significato in entrambi i mercati, accade in differenti situazioni.

Forex Slippage

Nel forex, lo slippage accade quando un ordine viene eseguito, spesso senza un limit order, o uno stop loss viene attivato ad un prezzo meno favorevole rispetto a quello aspettato[3]. Questo succede spesso nelle situazioni di alta volatilità, magari dovuta da news rilevanti sul mercato, risultando in un ordine impossibile da eseguire al prezzo desiderato; in questa situazione, la maggior parte degli operatori finanziari eseguono l'operazione al miglior prezzo disponibile successivo, a meno che la presenza di un limit order non chiuda l'operazione ad un predeterminato prezzo. Mentre un limit order può prevenire uno slippage negativo, questo si porta dietro il rischio intrinseco che l'operazioni non venga completamente eseguita se il prezzo non restituisce una quantità favorevole dell'asset in considerazione. Questo rischio aumenta in situazioni dove le fluttuazioni di mercato avvengono molto rapidamente, limitando significativamente la quantità di tempo disponibile per l'esecuzione di un trade ad un prezzo accettabile.

[3] www.investopedia.com/terms/s/slippage.

Stock Trading Slippage

Lo slippage, nel mercato azionario, accade spesso nel momento in cui c'è una modifica dello spread *bid-ask*. In questa situazione, un ordine a mercato[4] piazzato da un investitore potrebbe essere eseguito ad un prezzo meno favorevole rispetto a quello atteso; nel caso di un operazione *long*, il prezzo bid che viene eseguito potrebbe essere più basso rispetto alle aspettative. I traders possono proteggersi dal rischio dello slippage evitando di inserire ordini a mercato non necessari.

[4] Un ordine a mercato è un'istruzione con la quale viene eseguito un ordine al miglior prezzo disponibile al momento. Gli ordini a mercato sono i più semplici che esistono e non prevedono la definizione di un prezzo o di un periodo di tempo per l'esecuzione. Per questo motivo vengono di solito eseguiti molto velocemente (purché ci sia abbastanza liquidità nel mercato.) e, a volte, hanno commissioni inferiori rispetto ad altri tipi di ordine [38]

Capitolo 4

Project `At0m`: Atomic Arbitrage Bot

Dopo aver esaminato, nei capitoli precedenti, i concetti e gli strumenti fondamentali su cui basa il suo funzionamento, verrà presentato in questo capitolo `At0m`, un software ideato e sviluppato negli ultimi due anni in collaborazione con un software developer[1]. Tale software può essere descritto come un sistema automatizzato di investimento che permette di sfruttare le occasioni di arbitraggio presenti sui DExs (mercati decentralizzati, Sezione 2.13), grazie all'utilizzo di un sistema di smart contract presente sulla piattaforma Ethereum, avvalendosi inoltre ad una continua analisi dei dati provenienti dal mercato. La componente innovativa del progetto, spiegata più nel dettaglio nel Sezione 4.1, sta nell'esecuzione delle operazioni di arbitraggio, la quale permette di eliminare completamente la componente di rischio data dallo slippage del prezzo; questo è permesso grazie alla possibilità, sulla blockchain Ethereum, di inserire operazioni multiple (come l'acquisto e la vendita di un asset) all'interno della stessa, unica, transazione.

[1] William Bergamo, ex studente della facoltà di informatica dell'Università Ca' Foscari e software developer per diverse start-up dell'incubatore H-Farm.

4.1 Arbitraggio atomico

Continuando ad analizzare quanto presentato nel precedente paragrafo, l'arbitraggio può essere considerata un'attività redditizia, seppur limitata da vari fattori e influenzata, nella sua applicazione reale, da una serie di rischi, trattati nella Sezione 3.6 del Capitolo 3.

Uno di questi rischi, lo slippage (Sezione 3.6.4), può essere completamente eliminato dall'operazione di arbitraggio grazie all'utilizzo degli smart contracts. Le opportunità che si presentano grazie all'arbitraggio tramite smart contract hanno una caratteristica aggiuntiva, che le distingue da quelle presenti nelle varie tipologie di arbitraggio *cross-exchange* analizzate nel Capitolo 3.

Come già trattato nel Paragrafo 2.9 del Capitolo 2, grazie all'elaborazione *atomica*[2] delle transazioni nella EVM, basata sull'esecuzione di un *batch* di transazioni (quindi un'insieme di transazioni), e grazie al fatto che le transazioni possono essere attivate tramite uno smart contract, è possibile sviluppare *bot*[3] che eseguono delle operazioni di arbitraggio in diversi DExs attraverso uno smart contract. Questo contratto può essere definito un *proxy contract* [23], in quanto tale contratto permette la comunicazione fra i segnali proveniente dai bot nel momento in cui è presente un'occasione di arbitraggio e gli smart contracts dei mercati decentralizzati per eseguire l'effettiva operazione di arbitraggio.

Ciò significa che il bot per l'arbitraggio è in grado di comporre una singola transazione con al suo interno una serie di operazioni di scambio atomico di tokens, con un modello "*all-or-nothing*"; questo permette di eliminare completamente il rischio di slippage, in quanto l'operazione viene completata solamente nel caso in cui l'opportunità di

[2] Un'operazione atomica consiste nell'esecuzione di un'operazione indivisibile dal punto di vista logico. In generale, un'operazione si dice atomica se è indivisibile, ovvero se nessun'altra operazione può cominciare prima che la prima sia finita. Il risultato di quella operazione è sempre lo stesso se parte dalle stesse condizioni iniziali.

[3] Un bot è uno script che esegue automaticamente una serie di operazioni nel momento in cui si verificano determinate condizioni.

arbitraggio sia effettivamente eseguibile. Nel caso in cui ci si ritrovi in una situazione di non-arbitraggio, la transazione verrà annullata e l'unica perdita sarebbe rappresentata dal prodotto fra `gasPrice` e `gasLimit`, utilizzati come parametri (come descritto nella Sezione 2.6) della transazione.

Si consideri tale esempio: una transazione comprende al suo interno l'acquisto di un asset al prezzo x e la sequenziale vendita ad un prezzo $x' > x$; se eseguita atomicamente, queste operazioni generano un profitto garantito. Ad esempio, uno smart contract potrebbe eseguire un'operazione di acquisto di un token ERC20[4] per 2 ETH[5], e un'operazione di vendita al prezzo di 3 ETH. Se gli ordini sono concretamente eseguibili in entrambi i DExs, lo smart contract può eseguire entrambe le operazioni in un'unica transazione ed ottenere un profitto da arbitraggio pari a 1 ETH.

Nell'arbitraggio tradizionale, le operazioni devono essere analizzate probabilisticamente, in quanto vi è un'alta probabilità che solamente una delle due operazioni venga eseguita, interrompendo il processo di arbitraggio senza ottenere un profitto [23]. Questo rende l'arbitraggio basato sugli smart contracts un'operazione meno rischiosa e più semplice da osservare e analizzare rispetto all'arbitraggio *cross-exchange* tradizionale, in quanto le operazioni vengono eseguite in modo deterministico. É quindi possibile determinare a priori se l'arbitraggio verrà eseguito con successo, osservando le transazioni presenti sull'ultimo blocco che sta per essere validato.

4.2 Obiettivi di Project `At0m`

Gli obiettivi principali del progetto possono essere riassunti in:

- creazione di un sistema automatizzato di investimento che permetta di ottenere un rendimento passivo costante nel tempo;

[4] I token standard per la blockchain Ethereum, si veda Sezione 2.11.1
[5] *Ticker* di *ether*, valuta della blockchain Ethereum.

- eliminazione del rischio di slippage grazie all'inclusione delle due operazioni dell'arbitraggio in un unica transazione;

4.3 Architettura del bot

L'architettura di At0m è illustrata in Figura 4.1:

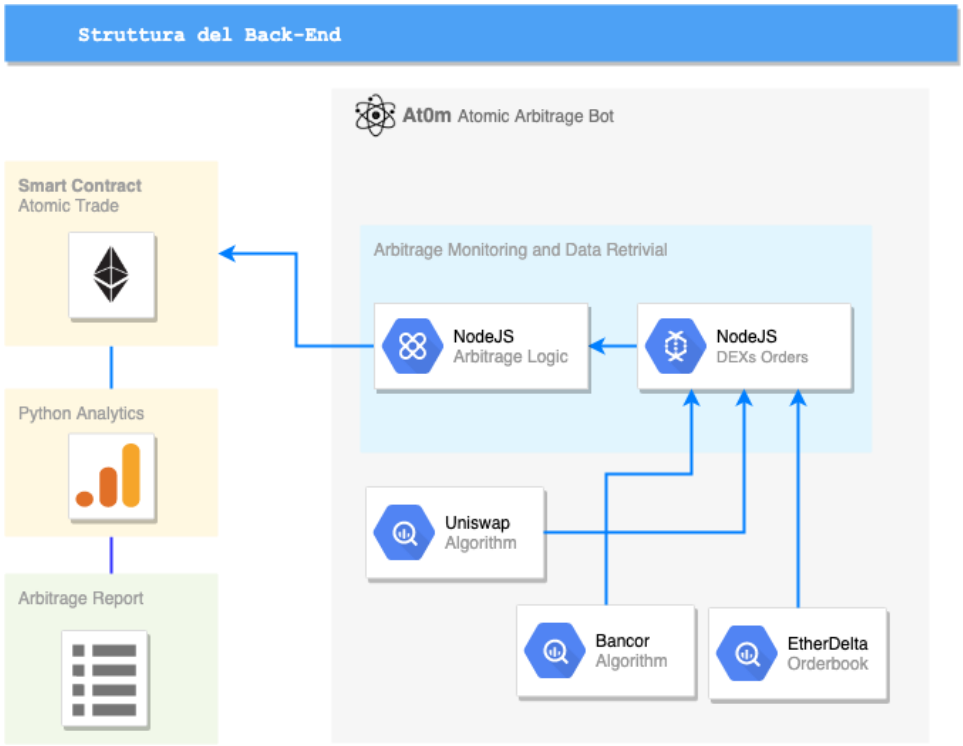

Figura 4.1: Architettura di At0m

Come è possibile osservare in Figura 4.1, sono presenti i moduli *Arbitrage Logic* e *DEXs Orders* che si occupano della gestionee dell'analisi dei flussi informativi provenienti dai DExs; le informazioni vengono inviate tramite delle richieste che vengono inviate agli smart contracts dei DEXs coinvolti (*Bancor*, *EtherDelta*, *Uniswap*). Ogni modulo è stato sviluppato *ad-hoc* per la piattaforma da cui deve estrarre i dati relativi

all'orderbook ed ai prezzi dei tokens. Il backend[6] di `At0m` viene eseguito su una VPS[7], la quale permette di avere il bot[8] operativo ininterrottamente e al contempo permette di avere i dati relativi alla blockchain perfettamente aggiornati, senza doversi affidare ad una terza parte per ottenere tali informazioni[9]. Le informazioni relative ai prezzi degli assets vengono analizzate e confrontate nel modulo centrale, dove avviene l'elaborazione dei dati che permette di individuare l'esistenza di occasioni di arbitraggio e, nel caso in cui vengano individuate delle discrepanze di prezzo fra diversi DExs, avviare la transazione di arbitraggio atomico.

[Le informazioni generate dal modulo ArbitrageLogic vengono inviate tramite una tx al uno smart contract (principale) che poi smista ad altri smart contracts]

Infine, viene effettuata una chiamata allo smart contract, il quale esegue un ulteriore controllo per verificare l'esistenza dell'occasione di arbitraggio e gestisce le ulteriori chiamate agli smart contracts dei DExs per eseguire l'acquisto e la conseguente rivendita dell'asset preso in considerazione. Per conludere l'analisi sull'architettura del bot, è presente uno script Python che estrae i dati dalla blockchain e li riorganizza per monitorare le prestazioni e risultati di `At0m`.

L'algoritmo utilizzato dal bot è ispirato all'arbitraggio bilaterale puro, descritto nel Capitolo 3. In particolare, ci si trova in una situazione di arbitraggio in presenza di uno spread bid-ask, e bisogna inoltre conteggiare l'incidenza del costo del gas[10] sul calcolo del profitto netto proveniente dalle operazioni di arbitraggio.

---

[6] I termini frontend e backend denotano, rispettivamente, la parte visibile all'utente e con cui egli può interagire (interfaccia utente) e la parte che permette l'effettivo funzionamento di queste interazioni.

[7] In ambito informatico, con virtual private server (comunemente VPS) ci si riferisce ad una macchina virtuale che permette di eseguire gli stessi compiti di un sever. Più VPS possono essere eseguiti contemporaneamente sullo stesso hardware (host computer); in tal modo è possibile ospitare più sistemi in esecuzione, anche con sistemi operativi differenti, potendo così risparmiare sui costi delle infrastrutture.

[8] Un bot, in terminologia informatica, è un programma che è esegue una serie di attività attività automatizzate (lo script). In genere, i bot eseguono compiti che sono sia semplici che strutturalmente ripetitivi.

[9] Ad esempio Infura, un provider di nodi Ethereum che consente agli utenti di eseguire la propria applicazione senza necessariamente configurare il proprio nodo o wallet Ethereum.

[10] Il gas è il costo di commissione pagato per eseguire una transazione sulla blockchain Ethereum

Il profitto può essere calcolato, ad esempio (considerando un operazione di arbitraggio che coinvolge Bancor ed EtherDelta, due DExs che verranno analizzati più nel dettaglio nella Sezione 4.4) nel seguente modo

$$\pi = S_{b,Bancor} - S_{a,EtherDleta} - gasfee \tag{4.1}$$

oppure

$$\pi = S_{b,EtherDelta} - S_{a,Bancor} - gasfee \tag{4.2}$$

L'operazione di arbitraggio risulterà profittevole nel momento in cui $\pi > 0$. Il bot verrà azionato solamente nel momento in cui verranno individuati un prezzo bid e un prezzo ask che permettano di avere un profitto maggiore delle commissioni per i miners. Si può inoltre affermare che l'arbitraggio risulterà profittevole se

$$S_{a,EtherDelta} < S_{b,Bancor} - gasfee \tag{4.3}$$

oppure

$$S_{a,Bancor} > S_{b,EtherDelta} - gasfee \tag{4.4}$$

La rappresentazione grafica delle relazioni (4.3) e (4.4) può essere rappresentata dal Grafico 4.2

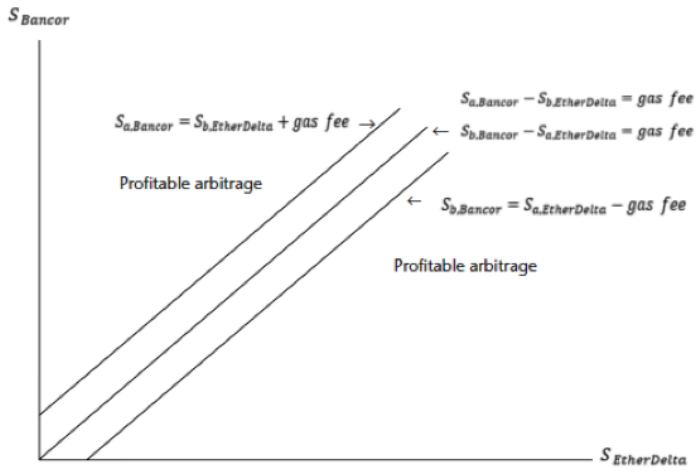

Figura 4.2: Linea di non-arbitraggio atomico

I punti che ricadono al di sopra e al di sotto delle due rette rappresentate nel grafico definiscono delle occasioni profittevoli di arbitraggio. Al di sopra del limite superiore, come trattato nel Capitolo 3, l'arbitraggio sarà profittevole comprando l'asset in Bancor per poi rivenderlo in EtherDelta, al netto del gas pagato nella transazione; al contrario, il limite inferiore rappresenta la soglia per le occasioni di arbitraggio profittevoli comprando in EtherDelta per poi rivendere in Bancor, al netto, come in precedenza, del gas.

4.3.1 Rischi operativi di `AtOm`

Nel caso in cui, nonostate tutti i controlli effettuati, e poi dal suo smart contract, l'operazione di arbitraggio non andasse a buon fine, si avrebbe come risultato quello della perdita totale delle commissioni utilizzate per il tentativo di esecuzione dell'operazione di arbitraggio. Questa comporta una perdita che può variare entro un range che va da 10cent a circa $1, come potrà essere osservato nell'analisi dei dati provenienti dalle operazioni di arbitraggio effettuate negli ultimi due mesi. Questo permette di predeter-

minare, prima di ogni transazione coinvolta in un operazione di arbitraggio, la possibile perdita nel caso in cui la transazione non venisse validata dai miners e l'arbitraggio non andasse quindi a buon fine. Tale possibilità non viene data, invece, agli arbitraggisti che operano sui mercati finanziari tradizionali, dove la perdita è proporzionale al capitale investito nell'operazione di arbitraggio.

4.3.2 Principali strumenti e librerie utilizzate

Verranno presentati in questa sezione gli strumenti e le principali librerie utilizzate per lo sviluppo di At0m, i quali permettono al bot di gestire le informazioni relative ai prezzi degli asset e di analizzarle per verificare l'esistenza di occasioni di arbitraggio.

Node.js

Node.js è un runtime environment[11] open-source e cross-platform per realizzare applicazioni Web in JavaScript, permettendo di utilizzare questo linguaggio, tipicamente "client-side", anche per la scrittura di applicazioni "server-side". La principale caratteristica di Node sta nel fatto che è basato sugli eventi e viene eseguito in modo asincrono. Le applicazioni sviluppate utilizzando a Node non seguono il tradizionale modello *receive, process, send, wait, receive*; ma processa le richieste in entrata (provenienti da un processo arbitrario) in uno stack di eventi, e può inviare a sua volta delle richieste senza dover attendere una risposta. Questo tipo di ambiente è stato utilizzato per gestire l'architettura logica del bot, per gestire l'analisi degli orderbook e la comunicazione con lo smart contract.

BigNumber.js

Come viene definita nella sua documentazione, BigNumber.js è

[11] Run-time environment (o runtime system) è il termine con cui in informatica si designa quel software che fornisce i servizi necessari all'esecuzione di un programma, pur non facendo parte in senso stretto del sistema operativo.

> "A JavaScript library for arbitrary-precision decimal and non-decimal arithmetic."

Permette infatti di eseguire una serie di operazioni aritmetiche minimizzando la possibilità del *floating point problem*[12] Questa libreria diventa utile nel momento in cui è necessario verificare l'esistenza di un'opportunità di arbitraggio: ether e i token ERC20 possono arrivare fino a 18 decimali, quindi è necessario utilizzare una libreria dedicata alla gestione delle cifre decimali per non produrre dei segnali errati, in particolare nel momento in cui è necessario rilevare l'esistenza di un'opportunità di arbitraggio.

OpenZeppelin

OpenZeppelin è un framework[13] per programmare smart contracts sicuri su Ethereum. Con OpenZeppelin, è possibile creare applicazioni e protocolli decentralizzati utilizzando pattern comuni di sicurezza dei contratti e il linguaggio di programmazione Solidity.

I principali vantaggi del framework OpenZeppelin:

- Focalizzato sulla sicurezza degli smart contracts
- Approccio modulare
- Codice semplice che permette una facile collaborazione e revisione
- Open source

In `At0m`, OpenZeppelin è stato utilizzato per lo sviluppo della gestione delle transazioni che vengono eseguite dallo smart contract.

[12] Il *floating point problem* è un problema, presente nei computers, che causa un errato arrotondamento delle cifre decimali nel momento in cui il calcolo viene eseguito; questo è dovuto dall'architettura binaria su cui i processori sono basati.

[13] Un framework, termine della lingua inglese che può essere tradotto come intelaiatura o struttura, in informatica e specificatamente nello sviluppo software, è un'architettura logica di supporto (spesso un'implementazione logica di un particolare *design pattern*) su cui un software può essere progettato e realizzato, spesso facilitandone lo sviluppo da parte del programmatore.

web3

La libreria web3.js è una raccolta di moduli che contengono funzionalità specifiche per l'ecosistema Ethereum. Web3-eth viene utilizzato per interagire con la blockchain ethereum e per gli smart contracts. Il web3-utils contiene funzioni di supporto per gli sviluppatori di DApp. Tramite web3 è possibile interagire con la blockchain di Ethereum, quindi chiamare funzione contratti leggere dati dalla blockchain, leggere e mandare transazioni via questa libreria Javascript scrivere che si appoggia al nostro nodo o usa servizio (in generale deve connettersi a qualcosa)

sviluppare *patters* di programmazione che permettono di eseguire funzioni asincrone; inoltre, in web3.js, è possibile effettuare *callbacks*[14], o utilizzare una logica a promesse[15] per concatenare più funzioni [39]. Questo in quanto, in Ethereum, una transazione può assumere diverse forme ("inserita nel blocco", "non valida", "eseguita", . . .) e quindi ha bisogno per restituire più "stadi" di una determinata azione. Tale libreria permette quindi di interagire con gli smart contracts dei DExs, consentendo di ricevere le informazioni riguardanti i prezzi, gli ordini presenti nell'orderbook e il wallet.

4.4 Mercati decentralizzati (DExs) utilizzati

4.4.1 Bancor Protocol

Bancor Protocol è un protocollo open-source per la determinazone automatica dei prezzi e che permette il funzionamento di un meccanismo autonomo di liquidità, il

[14] In programmazione, un callback (o, in italiano, richiamo) è, in genere, una funzione, o un "blocco di codice" che viene passata come parametro ad un'altra funzione. In particolare, quando ci si riferisce alla callback richiamata da una funzione, la callback viene passata come argomento ad un parametro della funzione chiamante. In questo modo la funzione chiamante può realizzare un compito specifico (quello svolto dalla callback) che non è, molto spesso, noto al momento della scrittura del codice.

[15] In informatica, con il termine promise ci si riferisce a un particolare costrutto usato nella sincronizzazione dell'esecuzione dei programmi scritti con alcuni linguaggi di programmazione di tipo concorrente. Questo costrutto fa uso di una sorta di segnaposto, un oggetto che prende il posto di un valore che è in attesa di essere restituito da un'operazione asincrona. Su questo oggetto vengono definite delle operazioni da effettuare quando l'operazione asincrona sarà stata completata.

quale consente di poter convertire costantemente i vari tokens presenti nel network, ad un prezzo calcolato tramite un algoritmo. Bancor non permette lo scambio peer-to-peer fra gli utenti della piattaforma, ma piuttosto permette agli utenti di convertire i propri fondi in tokens che comunicano direttamente con lo smart contract di Bancor, con la possibilità di convertirli a loro volta in altri token in ogni momento con un prezzo calcolato istantaneamente [40]. Lo smart contract monitora le riserve disponibili di tokens e funge da controparte per ogni trade. Il contratto di Bancor si occupa anche di aggiornare automaticamente le informazioni riguardanti i tokens, come il prezzo e la quantità disponibile in tutto il network, basandosi su un algoritmo pubblico. Nello specifico, Bancor permette ai suoi utenti di convertire ether o tokens ERC-20 in altri tokens utilizzando il meccanismo degli Smart Tokens: questi particolari tokens sono convertibili in altri tokens del network ad un prezzo calcolato algoritmicamente utilizzando le riseve di tokens disponibili [41].

L'algoritmo permette di prevenire la manipolazione dei prezzi e dei volumi, pratica comune nei tradizionali sistemi di matching degli ordini [42]. Dalla prospettiva degli utenti, scambiare un asset con uno smart contract è funzionalmente uguale a scambiare un asset con un altro utente. Uno Smart Token permette, ad esempio, di convertire il token base di Bancor (*BNT*) con qualsiasi altro token denetuto dallo smart token (utilizzeremo *ETH* per l'esempio). L'utente dovrà semplicemente inviare *BNT* (o *ETH*) allo smart contract e riceverà *ETH* (o *BNT*) ad un tasso di conversione che permette allo Smart Token di mantenere un rapporto predeterminato fra le riserve dei due tokens coinvolti nello scambio. Nel caso in cui lo smart token sia configurato per mantenere un rapporto del 10% *BNT*:*ETH*, quando gli utenti spediranno dei *BNT* allo smart contract per ottenere in cambio degli *ETH*, il prezzo di *BNT* in termini di *ETH* verrà aggiustato dinamicamente per mantere questo rapporto fra le due pool di liquidità.

Per un'analisi più approfondita di Bancor Protocol, del suo ecosistema e dei suoi principali vantaggi, si veda l'Appendice C Sezione C.2.

4.4.2 Uniswap

Uniswap è un protocollo open-source per lo scambio automatizzato di token su Ethereum. È stato progettato attorno ad una serie di principi base:

- facilità d'utilizzo;
- efficienza in termini di costo del gas (costo delle commissioni).

Tale protocollo può essere utilizzato sia da trader, per scambiare tokens, che da smart contracts, per automatizzare delle operazioni che necessitano una fonte di liquidità *on-chain*, come per l'arbitraggio.

Gli smart contracts di Uniswap contengono riserve di liquidità di vari token e le negoziazioni sono eseguite direttamente contro queste riserve. I prezzi vengono determinati automaticamente utilizzando il cosiddetto meccanismo del *constant market maker* ($x \times y = k$) [43], che mantiene le riserve complessive di tokens in un equilibrio relativo; le riserve vengono formate inizialmente da una rete di fornitori di liquidità che forniscono tokens al sistema in cambio di una quota proporzionale delle commissioni sulle transazioni.

Una caratteristica importante di Uniswap è l'utilizzo di un contratto intermedio (*factory/registry contract*), il quale implementa uno smart contract separato per ciascun token ERC20. Questi contratti contengono contengono le riserve di ETH e del relativo token ERC20. Ciò consente di scambiare le due valute sulla base dell'offerta relativa disponibile. I vari smart contract specifici sono collegati fra di loro tramite il contratto intermedio, consentendo scambi diretti tra ERC20 e ERC20 o tra qualsiasi altro token che utilizza ETH come un intermediario [43].

4.4.3 EtherDelta

EtherDelta è un DEx che permette ai suoi utenti di scambiare token ERC20, utilizzando la modalità del peer-to-peer, con altri utenti. La piattaforma utilizza un orderbook

"tradizionale" che permette la visualizzazione dei migliori 500 ordini di acquito e di vendita per ogni asset quotato, ordinati per prezzo e dimensione.

Gli utenti interagiscono con EtherDelta sia come maker che come taker; i primi inseriscono limit orders per comprare o vendere un token specificandone sia il prezzo che la quantità, con la possibiltà di stabilire anche una scadenza temporale per l'ordine, dopo la quale esso verrà cancellato. Quando un maker firma e inserisce una transazione nella piattaforma, l'ordine viene aggiunto nell'orderbook di EtherDelta, ospitato in uno dei server centralizzati e pubblicamente visibile nel sito web della piattaforma. Per quanto riguarda i takers invece, non fanno altro che eseguire gli ordini già esistenti nel orderbook selezionandone uno fra i primi 500 ordini inseriti dai makers. Attualmente, EtherDelta non supporta gli ordini a mercato.

La piattaforma in sé, infatti, non performa il *matching* degli ordini; invece, i takers devono selezionare degli ordini specifici presenti nell'ordebook; una volta individuato l'ordine desiderato, la piattaforma invia sia l'ordine maker che l'ordine taker allo smart contract di EtherDelta per la loro validazione e la conseguente esecuzione, per poi essere inseriti nella blockchain Ethereum. Una volta ottenuta la conferma di questo inserimento, che avviene on-chain, l'orderbook di EtherDelta viene aggiornato; i tokens relativi all'operazione appena conclusa fra il maker e il taker vengono trasferiti direttamente da un wallet all'altro.

4.5 Risultati di Project At0m

Verranno ora riportati i risultati relativi alle operazioni di arbitraggio effettuate da At0m. La fonte di tali informazioni è *etherscan.io*; tale piattaforma è il principale BlockExplorer per la blockchain Ethereum. I dati sono relativi al periodo *26 Novembre 2018 - 13 Giugno 2019*; il capitale iniziale è pari a 3,9 ETH (circa $500 al cambio storico, dicembre 2018).

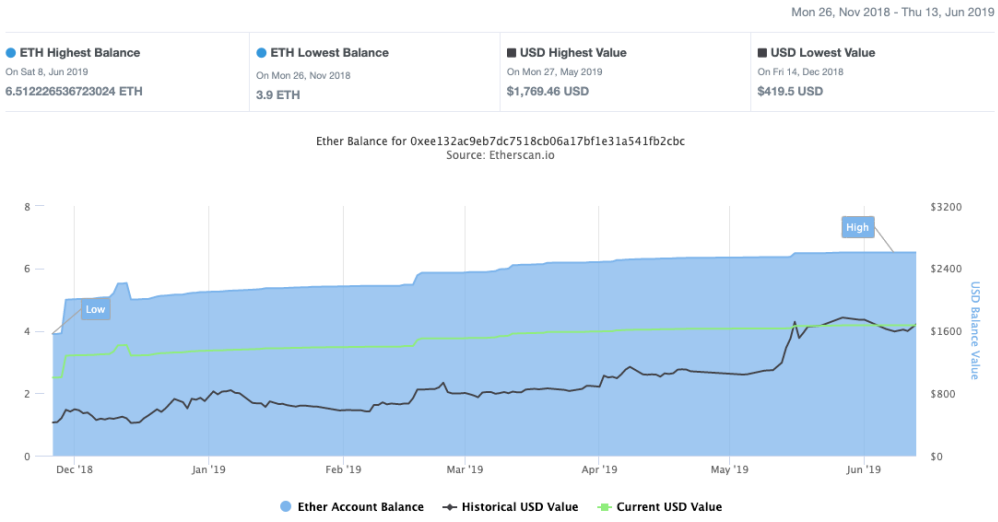

Figura 4.3: Riepilogo dell'andamento del wallet di At0m

Tabella 4.1: Riepilogo dell'andamento dei profitti di At0m (1 ETH = $268.12, 20/06/2019)

Month	Arbitrage Profit (ETH)	Arbitrage Profit ($)	Marginal Profit per TX
Dec 2018	0.9388024488	$252.53	$5.15
Jan 2019	0.06221778184	$16.64	$0.43
Feb 2019	0.373794127	$100.55	$6.70
Mar 2019	0.2767815946	$74.45	$1.77
Apr 2019	0.07460584614	$20.07	$1.00
May 2019	0.09950389142	$26.77	$1.57

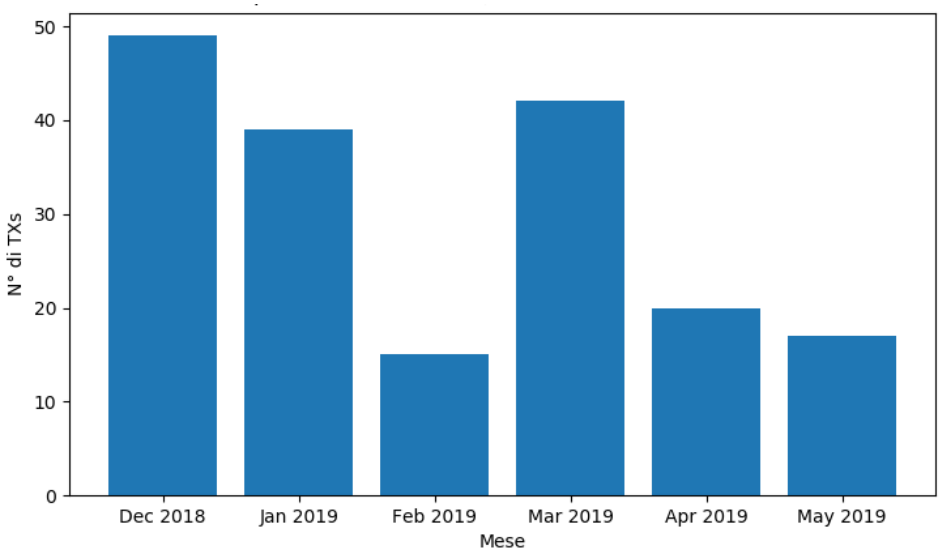

Figura 4.4: Panoramica sulle operazioni di arbitraggio avvenute con successo

L'attività di At0m ha portato ad un profitto pari a 1,827910375 ETH, grazie alle operazioni di arbitraggio avvenute con successo; tuttavia, conteggiando anche le transazioni fallite e i loro relativi costi di commissione, il profitto netto diventa 0.7196759612 ETH. Questo dimostra quanto sia importante la gestione del gas nella pratica dell'arbitraggio atomico; un'attenta gestione del gas permetterebbe un'ottimizzazione dei profitti, rendendo il bot ancora più profittevole.

Il motivo principale per cui una transazione fallisce è un errata impostazione del `gasLimit`. Ciò può portare infatti a non includere la transazione nel blocco che viene validato dai miners, con la conseguente perdita delle commissioni pagate. Questo succede per:

- una reward per i miners non sufficiente;
- frontrunning da parte di altri arbitraggisti [23].

Il frontrunning è una pratica che generalmente sfrutta le asimmetrie informative create da alcuni soggetti presenti all'interno di una struttura finanziaria (ad esempio, gli intermediari che hanno accesso privilegiato alle informazioni relative ai mercati).

In quanto questa figura non è presente in un sistema decentralizzato, le asimmetrie informative possono sorgere per gli attori che hanno il *know-how* necessario per accedere ed interpretare l'infrastruttura sottostante, cioè la blockchain [23].

4.5.1 Esempio di una transazione di arbitraggio atomico

Di seguito verrà riportato un esempio di una transazione di arbitraggio atomico effettuata da `At0m` in data 18 Febbraio 2019. L'hash della transazione è

`0xc5bbeedde1a2f49a780d3dacf384d1eb8aceaafafaf2e2bc1b92bb364de5c768`

ed è possibile reperire tali informazioni presso *etherscan.io*.

Il capitale iniziale dell'operazione è $1,3742103 ETH$, e i DExs coinvolti sono EtherDelta (*acquisto iniziale*) e Bancor (*rivendita finale*). La Figura 4.6 mostra nel dettaglio gli scambi di tokens avvenuti all'interno dell'operazione di arbitraggio. Si può notare che il primo scambio permette di ricevere $4567 SHP$ (Sharpe token, un token ERC20); il prezzo dell'asset sarà quindi:

$$S_{a,EtherDelta} = 0,0003009 ETH \tag{4.5}$$

I successivi scambi sono necessari per convertire il token ERC20 in uno smart token di Bancor, per poi rivenderlo ad un prezzo:

$$S_{b,Bancor} = 0,00036679 ETH \tag{4.6}$$

Utilizzando quindi l'Equazione (4.3), è possibile verificare l'effettiva esistenza di un'opportunità di arbitraggio. Tenendo presente che

$$\texttt{gasPrice} = 0,000000011 ETH$$

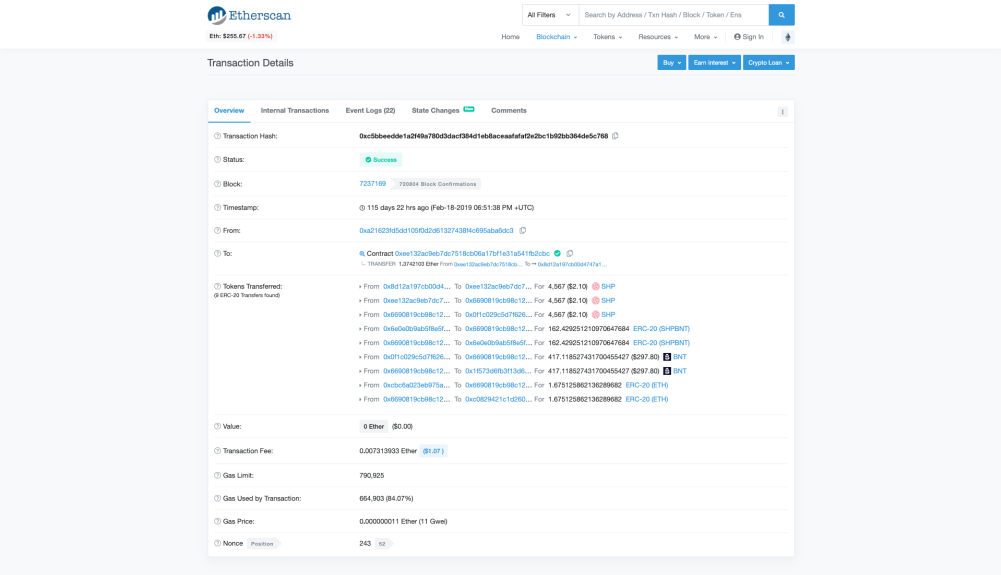

Figura 4.5: Informazioni generali sull'operazione di arbitraggio atomico

si può dimostrare che la relazione (4.3) viene rispettata:

$$0,0003009ETH < 0,00036679ETH - 0,000000011ETH \tag{4.7}$$

Figura 4.6: Dettaglio dei tokens scambiati nell'operazione di arbitraggio atomico

Si può infine notare che la rivendita finale del token ERC20 in Bancor ha permesso di ricevere $1,67512586ETH$, come è possibile osservare nella figura 4.6, ottenendo un profitto pari a $0,30091556$, che al netto del costo di transazione ($0.00731393ETH$) diventa pari a $0,29360162ETH$, cioè circa \$75 ($1ETH = \$255,67$, quotazione del 14/06/2019).

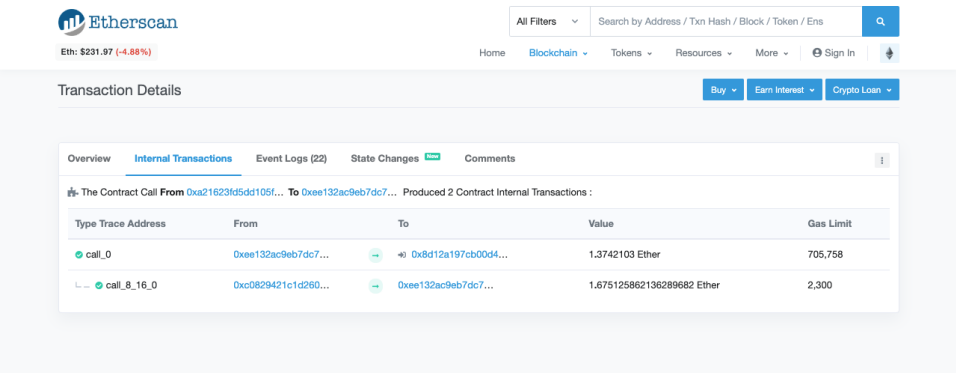

Figura 4.7: Transazioni interne dell'operazione di arbitraggio atomico

Capitolo 5

Conclusioni

La tecnologia blockchain, in particolare Ethereum, sta attirando molta attenzione non solo per l'aspetto speculativo dell'investimento in cryptovalute; ma, dall'altra parte, per la massiccia moltitudine di possibilità offerte dalle sue proprietà, dagli smart contracts e dalle dApps. A mio parere, il vero impatto della blockchain deve ancora venire perché, e anche se la tecnologia è già utilizzabile, occorreranno molti anni per penetrare nella mentalità delle persone, così come lo è stata per Internet.

L'obiettivo di questa tesi, grazie ad `AtOm`, è stato raggiunto: è possibile, utilizzando le logiche presenti sulla blockchain di Ethereum, eliminare lo slippage dall'operazione di arbitraggio. Questo può, in parte, dimostrare come l'introduzione di nuove tecnologie possa permettere di ottimizzare delle pratiche già presenti nella finanza tradizionale. Nonostante i possibili miglioramenti e le inefficienze da eliminare siano molteplici, l'entrata in scena di una nuova branca della finanza, chiamata *Open Finance* o *Decentralized Finance* [44], può dimostrare come la blockchain possa rivoluzionare un settore tradizionale come quello della finanza.

Open Finance fa riferimento a una serie di protocolli decentralizzati che creano un'infrastruttura finanziaria open-source. Questi protocolli sono fondamentali in quanto stanno creando le fondamenta necessarie per consentire a chiunque nel mondo con

una connessione Internet di accedere a servizi finanziari auto-gestiti e resistenti alla censura.

Nel sistema attuale, tutti i servizi finanziari sono controllati da una parte centrale. Che si tratti di trasferimenti di denaro, acquisti di assets o prestiti, è necessario rivolgersi a un intermediario che addebita una commissione per la mediazione delle transazioni finanziarie. I servizi finanziari basati su Ethereum, invece, collegano le persone in modo peer-to-peer, consentono loro di accedere a vari strumenti finanziari più facilmente e in modo più economico.

Il principale motivo per cui il settore dei servizi finanziari è attualmente basato sulla mediazione riguarda il fatto che le transazioni finanziarie nel mondo digitale non sono sicure. Non esisteva alcun metodo per fidarci delle nostre controparti online, pertanto vengono pagate le commissioni agli istituti finanziari che permettono di fidarci delle transazioni.

Ethereum sfrutta gli stessi principi che creano la stessa "fiducia digitale" all'interno di Bitcoin e li applica a gli smart contracts, pezzi di codice auto-eseguibili che eseguono una serie di operazioni logiche dopo che sono state soddisfatte specifiche condizioni predefinite. Gli smart contracts assomigliano molto ai contratti finanziari in quanto forniscono fondi a terzi e li trasferiscono in risposta a determinati eventi. Questo sistema può essere considerato superiore in quanto la logica di business codificata non può essere manipolata da una parte centrale, una volta che è stata implementata nella rete principale di Ethereum.

I servizi finanziari basati su Blockchain possono inoltre essere considerati superiori alle loro controparti centralizzate in base alle seguenti caratteristiche:

- *Permissionless*: una connessione Internet è tutto ciò che serve per accedere a questi servizi;

- *Censorship Resistance*: nessuna parte centrale è in grado di modificare l'ordine e i dettagli delle transazioni e disattivare i servizi;
- *Trasparent*: una blockchain pubblica come Ethereum è completamente trasparenti e verificabili;
- *Programmable*: gli sviluppatori possono creare e intrecciare servizi finanziari a un costo molto basso;
- *Efficiency*: i servizi finanziari open-source sono alimentati dal codice, non dagli umani, e in quanto tali comportano costi di middleman molto inferiori, se non del tutto eliminati.

Per quanto riguarda, invece, l'implementazione di nuove caratteristiche per `At0m` può essere descritta in diverse fasi, separate l'una dall'altra per permettere agli sviluppatori di testare ed affinare al meglio prima di rendere pubbliche tali features. Le fasi sono:

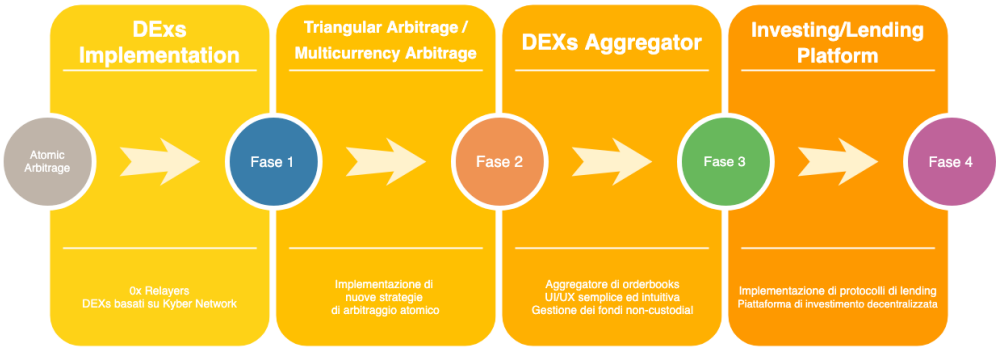

Figura 5.1: Rappresentazione delle fasi di sviluppo di `At0m`

Fase 1

Come è possibile osservare in Figura 5.1, la fase iniziale di sviluppo prevede l'ampliamento del numero di piattaforme decentralizzate in cui `At0m` può operare, aumentando

così la possibilità di individure discrepanze di prezzo. Gli exchanges decentralizzati che possono essere implementati sono:

- Tutti i DEXs basati su 0x o Kyber Network.
- Tutti i DEXs basati su smart contracts che permettono l'interazione di smart contracts esterni.

Questo, oltre a migliorare il processo di *arbitrage discovery*, comporterà un impegno continuo nello sviluppo e nel mantenimento dell'infrastruttura che permetterà di collegare tutti i DExs coinvolti nelle operazioni di arbitraggio; di conseguenza, gli sviluppatori dovranno essere impegnati costantemente nel monitoraggio e nel testing di tale architettura, per garantire un suo continuo funzionamento e mantenimento.

Fase 2

La seguente fase prevede l'inserimento delle logiche di arbitraggio triangolare e di arbitraggio multi-valutario in `At0m`; una volta implementate, il bot avrà la possibilità di sfruttare occasioni di arbitraggio concatenate, aumentando il profitto proveniente da tali operazioni. La *fase 1* è propedeutica all'attuazione della *fase 2* in quanto è necessario avere a disposizione DExs multipli per poter attuare strategie di arbitraggio triangolare o multi-valutario. L'elemento critico per il successo di tale implementazione rimane sempre l'abilità di `At0m` a rispondere in modo veloce all'individuazione di un'opportunità di arbitraggio, indifferentemente che essa sia bilaterale, triangolare o multi-valutaria.

Fase 3

In questa fase, dopo aver raggiunto la capacità di interagire con un cospicuo numero di exchanges decentralizzati, si prevede di trasformare `At0m` in un vero e proprio aggregatore di ordini proveniente da altri DExs o protocolli decentralizzati. Questo grazie al

fatto che la struttura degli smart contracts necessaria al funzionamento dell'arbitraggio atomico può essere utilizzata anche per aggregare gli ordini presenti sugli orderbooks. La mancanza di liquidità è uno dei punti deboli più conosciuti sia dagli sviluppatori che dai traders dei DEXs, portando ad una forte frammentazione del mercato. L'evoluzione di `AtOm` prevede quindi la costituzione di una piattaforma che aggreghi gli ordini presenti nelle varie piattaforme decentralizzate per aumentare la limitata disponibilità di ordini. Aggregando in questo modo gli orderbooks frammentati dei vari DEXs, sale vertiginosamente le probabilità di poter eseguire un ordine inserito da un utente della piattaforma. La piattaforma dovrebbe prevedere l'implementazione sia di un'interfaccia grafica con cui l'utente possa interagire per l'inserimento o l'esecuzione degli ordini, oltre a poter gestire, con un approccio non-custodial, il proprio portfolio di assets, sia una struttura di API per permettere agli utenti business di programmare e gestire i propri strumenti finanziari decentralizzati in modo completamente automatizzato, con la possibilità di monitorarli attraverso una eventuale dashboard ad-hoc.
Le caratteristiche chiave di questa piattaforma saranno:

- L'aggregazione del maggior numero possibile di orderbook di altri DEXs;

- Una User Experience e User Interface (UX/UI) semplice e intuitiva che, pur fornendo gli strumenti necessari agli investitori più esperti (fra cui margin trading, stop loss orders, limit order), renda l'interazione con la piattaforma un'esperienza veloce e diretta, permettendo all'utente di effettuare trades o gestire i propri fondi in pochi click;

- Gestione non-custodial e trustless dei fondi presenti sulla piattaforma;

- Meccanismo di incrocio automatico degli ordini presenti sugli orderbooks ed esecuzione istantanea dei trades.

Fase 4

In quest'ultima fase degli sviluppi a breve termina si prevede di comprendere all'interno della piattaforma descritta nella fase 3 i protocolli, nati da poco nell'ecosistema blockchain [44], per i prestiti peer-to-peer. In tal modo, i fondi investiti, oltre ad essere fatti fruttare attraverso l'arbitraggio atomico, potrebbero percepire un interesse nel momento in cui vi sia una situazione di non-arbitraggio. La piattaforma dovrà quindi avere la dinamicità necessaria ad integrare i protocolli già esistenti e quelli che verranno sviluppati in futuro, per poter offrire ai suoi utenti una vasta gamma di prodotti e strumenti di investimento o finanziamento; tutto ciò senza nessuna intermediazione, e il tutto verrà garantito dalle leggi crittografiche intrinseche alla struttura della blockchain. La crescita dei protocolli su blockchain permetterà la costruzione di un ecosistema finanziario decentralizzato dove la facilità d'utilizzo dei vari strumenti giocherà un ruolo fondamentale.

Appendice A

Concetti e meccanismi del Bitcoin

In questa sezione descriveremo i concetti principali del Bitcoin, oltre ad alcuni dettagli sulle varie implementazioni. Come prima cryptovaluta e come valuta con capitalizzazione di mercato più alta (stando alla data odierna, Giugno 2019), è anche probabilmente la più documentata. La comprensione dei concetti chiave e delle implementazioni del Bitcoin può risultare utile per approfondire lo studio di altre cryptovalute come Ethereum.

A.1 Il ruolo della rete dei nodi (*node network*)

Bitcoin utilizzza un'architettura di rete distribuita peer-to-peer (P2P) e i nodi all'interno della rete comunicano utilizzando il protocollo TCP[1] con una porta default 8333. In generale, i nodi possono assumere quattro ruoli differenti per rendere sicuro e poter utilizzare il network Bitcoin. Questi compiti non si possono escludere a vicenda, diversamente dai nodi.

[1] In telecomunicazioni e informatica il Transmission Control Protocol (TCP) è un protocollo di rete a pacchetto di livello di trasporto, appartenente alla suite di protocolli Internet, che si occupa di controllo della trasmissione, ovvero rendere affidabile la comunicazione dati in rete tra mittente e destinatario

I possibili ruoli sono:

A.1.1 Routers

I nodi con funzione di routing aiutano gli altri nodi a scoprire nuovi nodi all'interno della rete, in un processo chiamato network discovery process. Implementano così un middleware[2] che permette di ricevere ed emettere transazioni di Bitcoin. La maggior parte dei software Bitcoin ha implementato tale funzionalità.

A.1.2 Wallets

I nodi con funzione di wallet invece permettono di creare e inoltrare nuove transazioni di Bitcoin, oltre che tenere traccia dei pagamenti avvenuti. Possono supportare anche altre funzionalità, come la firma digitale crittografata di un testo. I nodi wallet sono inclusi in software per Bitcoin come `Bitcoin Core` o `Electrum`.

A.1.3 Miners

I nodi con funzionalità di mining ricevono e convertono le nuove transazioni in strutture di dati chiamate blocchi. I miners competono uno contro l'altro per risolvere un problema crittografico, una sorta di puzzle che richiede un determinato ammontare di potenza computazionale, per trovare un valore random, detto *nonce*. Il *nonce* è un valore numerico casuale che viene inserito fra gli inputs della funzione crittografica che permette di ottenere il numero identificativo univoco del blocco da validare; i miners competono fra di loro nella ricerca di questo numero, in modo tale che l'*hash* prodotto, cioè il numero identicativo univoco, abbia determinate caratteristiche dettate dalla difficoltà attuale del blocco[3]. Attraverso una competizione con ricompensa, i nodi

[2] Un insieme di software che fungono da intermediari tra diverse applicazioni e componenti software

[3] La difficoltà del blocco è una misura che determina, in termini di *hashing power* necessario, la complessità per validare un blocco per i miners; questa difficoltà viene aggiornata periodicamente in modo tale da assicurare un tempo di circa 10 minuti per la validazione di un blocco.

riescono quindi a ottenere un consenso generale, proprio perché conviene a tutti, dato che la posta in gioco sono le rendite di signoraggio date dall'emissione di nuova moneta tramite il *mining*, come verrà spiegato successivamente. Anche se il wallet Bitcoin Core ha implementato la possibilità di minare Bitcoin con la CPU, questa pratica ormai non è più profittevole allo stato dell'arte nel 2018. Software come `CGminer` e `BFGminer` supportano mining con ASIC, acronimo di *Application Specific Integrated Circuits*, ovvero chip specializzati e costruiti a misura con il solo scopo di minare cryptovalute.

A.1.4 Full nodes (nodi completi)

I nodi catalogati come full blockchain nodes o full nodes (nodi completi) contengono la base di dati completa delle transazioni di Bitcoin. Possono anonimamente validare tutte le transazioni esistenti (www.bitcoin.org/features/validation); la dimensione della blockchain nel 2018 è cresciuta approssimativamente da circa *160 GB* a *210 GB* [45].

A.2 Indirizzi (addresses)

L'email è diventata un metodo popolare di scambio di informazioni e l'indirizzo *email* è un concetto ben conosciuto dagli utenti. Il formato dell'indirizzo email consisteva originalmente in un identificatore per l'utente e il nome dell'host (il fornitore del servizio email), separati dalla cosiddetta "chiocciola" @. Tale convenzione è stata poi modificata per rendere compatibili i nomi dei domini al Domain Name System (www.tools.ietf.org). In modo simile, l'indirizzo *Bitcoin* è un concetto ben conosciuto fra i suoi utilizzatori ed è spesso utilizzato per identificare la destinazione quando si sta trasferendo dei bitcoins. Tuttavia, l'indirizzo Bitcoin non identifica direttamente né un particolare utente o macchina, né un nodo della rete. Nella seguente sezione gli indirizzi Bitcoin verranno descritti più nel dettaglio.

A.2.1 La costruzione dell'indirizzo

L'indirizzo Bitcoin si presenta spesso agli utenti come una stringa alfanumerica (www.en.bitcoin.it). Qui di seguito vengono riportati cinque indirizzi Bitcoin generati dal sito bitaddress.org, un generatore di indirizzi Bitcoin opensource basato su JavaScript:

```
1Ag71eTr2TjtbojAkBfmVFkmGWC2Z1VPC8
1PXHCkCMS1BAFXWAUHfoUnVKkg2JREyB3h
1GgmytEkDRLEJffimupMv7B1mrvN9QHYH8
19PCeT6Y9gRGbq5cVB6uu7B7h7zAT11khZ
12imR6JKttY76RwrMmkrJZrCnKZ5zCLTYM
```

Il fatto che tutti e cinque gli indirizzi generati inizino con il numero "1" non è una coincidenza; per capire la ragione di ciò, verrà descritto parte del processo di generazione dell'indirizzo (descritto più nel dettaglio in seguito [46]:

```
HASH_RAW = RIPEMD160(SHA256(DATA))
ADDRESS = BASE58CHECK(VERSION_BYTE+HASH_RAW)
```

dove *RIPEMD160* [47] e *SHA256* [48] sono delle funzioni crittografiche di hash, cioè un algoritmo matematico che mappa dei dati di lunghezza arbitraria in una stringa di dimensione fissa chiamata valore di hash, ma anche identificata con il termine inglese *message digest* (o semplicemente *digest*). *BASE58CHECK* è una funzione di codifica che incorpora anche una funzione che genera un checksum [49], cioè una sequenza di bit che viene utilizzata per verificare l'integrità di un dato o di un messaggio. Il *VERSION_BYTE* è un parametro che contiene le informazione relative al tipo (cioè la versione) dei dati contenuti nel parametro *DATA*. Infine, il parametro DATA, il più importante, contiene i dati che permettono di costruire l'indirizzo Bitcoin. Il *VERSION_BYTE* viene concatenato con l'*HASH_RAW* prima che venga codificato; questo

spiega perché tutti gli indirizzi generati hanno "1" come primo carattere - hanno tutti lo stesso *VERSION_BYTE*.

A.2.2 Tipologie di indirizzi

Come descritto nel libro di Andreas Antonopoulos in "Mastering Bitcoin", esistono molti tipi (o anche versioni) di indirizzi, i quali presentano tutti un carattere iniziale diverso:

- Indirizzi P2PKH (Pay-to-Pub-Key-Hash, che iniziano con "1");
- Indirizzi P2SH (Pay-to-Script-Hash, che iniziano con "3");
- Indirizzi "testnet" P2PKH (Pay-to-PubkeyHash, che iniziano con "*m*" o "*n*");
- Indirizzi "testnet" P2SH (Pay-to-ScriptHash, che iniziano con "2") - Gli indirizzi "testnet" vengono utilizzati dagli sviluppatori o dai tester per sperimentare e debuggare i software senza utilizzare bitcoins reali.

Il parametro *DATA* può contenere informazioni di due tipi:

- La chiave pubblica ECDSA (negli indirizzi, sia testnet che mainnet[4], P2PKH);
- Lo script del programma (negli indirizzi, sia testnet che mainnet, P2SH).

L'indirizzo Bitcoin non rappresenta quindi una posizione geografica; invece può rappresentare sia l'hash della chiave pubblica o l'hash dello script. Nella prossima sezione verranno analizzate nello specifico le transazioni di Bitcoin.

A.3 Transazioni

Le transazioni Bitcoin rappresentano un trasferimento di valore. Chiunque nel network peer-to-peer di Bitcoin può creare una nuova transazione all'interno della rete

[4] Il nome "mainnet" è utilizzato per distinguere il network principale, cioè dove vengono registrate le transazioni avvenute realmente, da quello per testare gli applicativi in via di sviluppo.

utilizzando un qualsiasi nodo. La transazione viene poi verificata dal nodo quando viene ricevuta; il tutto deve avere la corretta sintassi, la corretta dimensione in byte, un valore non minore di zero e i bitcoins all'interno della transazione devono essere autorizzati per essere utilizzati e devono soddisfare ulteriori regole di protocollo [50]. Quando una transazione viene verificata con successo, il nodo la propaga agli altri nodi conosciuti; questo processo di verifica e propagazione si ripete per ogni nodo e per ogni altra transazione. Come effetto si ha che le transazioni vengono rilevate in pochi secondi da tutti i partecipanti alla rete Bitcoin. Questo però non è sufficiente per far verificare equamente e con sicurezza le nuove transazione dai nodi Bitcoin. Alcuni di questi potrebbero "spendere" due volte gli stessi bitcoins o potrebberoi propagare la stessa transazione due volte; il consenso per la validità delle transazioni, come anche il consenso per quanto riguarda la storicità di una transazione dev'essere assolutamente raggiunto. Nel Bitcoin, nessuna autorità centrale valida le transazioni e previene la creazione di criticità come il sopracitato "double-spending problem", come non può neanche decidere se una transazione è permessa o meno. Invece, il consenso distribuito[5] è raggiunto tramite il processo chiamato mining.

La struttura delle transazioni

Una transazione Bitcoin è una struttura di dati che racchiude le informazioni riguardo il trasferimento di bitcoins. Essa consiste in una serie di campi di dati (*data fields*) con una dimensione in byte variabile, come viene mostrato nella Tabella A.1.

[5] Per consenso distribuito si intende il momento in cui la maggior parte dei nodi di una rete sono in accordo sull'esattezza e la validità delle transazioni contenute in un blocco e dell'ordine in cui sono state inviate.

Tabella A.1: Struttura delle transazioni [46]

Dimensione	Campo
4 bytes	Version
1-9 bytes	Input Counter
Variabile	Inputs
1-9 bytes	Output Counter
Variabile	Outputs
4 bytes	Locktime

Il campo *Version* è incluso per avere retrocompatibilità nel caso in cui il formato delle transazioni cambi in futuro. L'*Input couter* e l'*Output couter* hanno il compito di registrare il numero di inputs o outputs della transazione, i quali verranno descritti nella prossima sezione. Il campo *Locktime* può contenere il timestamp della transazione, cioè una sequenza di caratteri che rappresentano la data e l'ora della transazione, che permette di registrare temporalmente ogni transazione che verrà inserita nella blockchain.

Inputs e Outputs

Gli inputs e gli outputs sono entrambi concetti e strutture di dati fondamentali per la rete Bitcoin. Rappresentano infatti un trasferimento di valore (in bitcoin tokens) e al loro interno contengono i dati necessari per riceve l'autorizzazione a spendere (e quindi muovere) bitcoins. Come viene mostrato nella figura sottostante, il trasferimento del valore viene realizzato connettendo un output ad un input di una nuova transazione. Il mescolamento (mixing) dei valori di input è utilizzato per suddividere virtualmente il valore in uno o multipli outputs.

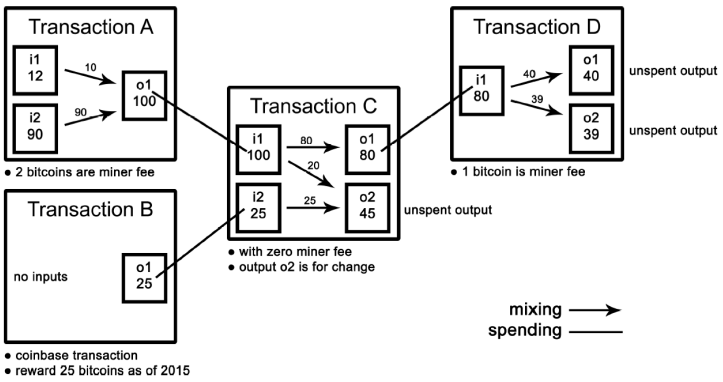

Figura A.1: Spending e mixing del valore (in bitcoin)

Output: Un output rappresenta e contiene il valore dei Bitcoin espresso in unità satoshi (1 bitcoin = 10^8 satoshi) come riportato nella Tabella A.2. Oltre a ciò, ogni output contiene le condizioni di autorizzazione codificate in un codice chiamato "locking script". Queste condizioni devono essere raggiunte per poter emettere l'output.

Tabella A.2: Inputs e Outputs

Dimensione	Campo
8 bytes	Valore (in satoshi)
1-9 bytes	Locking Script Size
Variabile	Locking Script

Input: Un input rappresenta il riferimento ad un output di un'altra transazione esistente. Tale riferimento rappresenta a sua volta il trasferimento di valore fra una transazione e l'altra - il cosiddetto *spending*. L'atto di "spendere" l'output comporta che quell'output diventa non più spendibile. Tuttavia, l'effettivo valore dell'output viene virtualmente spostato nella nuova transazione dove tale valore può essere mixato o diviso in nuovi outputs. In modo simile ai locking script degli output, un input contiene un codice chiamato "unlocking script"; quest'ultimo deve soddisfare le condizioni richieste dal locking script in un output.

Tabella A.3: Input

Dimensione	Campo
32 bytes	Hash della transazione precedente
4 bytes	Output Index
1-9 bytes	Dimensione dell'Unlocking Script
Variabile	Unlocking Script
4 bytes	Sequenza numerica

Commissioni: Come viene mostrato nella Figura A.1, se la somma di tutti gli outputs in una transazione è minore della somma di tutti gli inputs della stessa transazione ci sarà una differenza; questa non è altro che la commissione sulla transazione che viene data al miner che permette di inserire la transazione nella blockchain di Bitcoin. Esistono inoltre delle speciali transazioni chiamate "coinbase" che hanno un input di valore zero e nessun output. Tali transazioni emettono nuovi bitcoins nel sistema; possono essere create dai miners di Bitcoin come ricompensa per aver creato il consenso distribuito.

Script: Lo script, o anche scripting, è un linguaggio imperativo di programmazione stack-based[6] usato nei locking e unlocking scripts. Non è Turing-complete e quindi non supporta loops od operatori condizionali. Questa proprietà protegge i Bitcoin software da possibili loop infiniti [51] o attacchi malevoli che potrebbero rendere la rete Bitcoin meno efficiente in termini di velocità di validazione delle transazioni. Nel dettaglio, i programmi script sono "*stateless*", cioè tutti i dati richiesti dallo script devono essere inclusi nel programma e nessun dato può essere archiviato al di fuori del programma. Un semplice esempio di un programma scritto in linguaggio Script è riportato nella Tabella A.4.

```
( (2 + 5) == 8);
```

[6] Stack-based è un paradigma di programmazione che basa il suo funzionamento su un *main stack* (in italiano, una pila) a cui passare una serie di parametri per ottenere un determinato output.

Il programma Script viene scritto ed eseguito da sinistra verso destra e contiene tre informazioni (in questo caso numeri) e due comandi (chiamati "*opcodes*"):

Tabella A.4: Esecuzione dello script

Step 1	**Step 2**	**Step 3**	**Step 4**	**Step 5**
2	5	OP_ADD	8	OP_EQUAL

Il procedimento è il seguente:

1. Il numero 2 viene inserito nel "main stack" (cioè nello stack principale): `|2|;`
2. Il numero 5 viene inserito nel "main stack": `|2, 5|;`
3. Gli ultimi due oggetti inseriti vengono eliminati dallo stack e vengono rimpiazzati dalla loro somma: `|7|;`
4. Il numero 8 viene inserito nel "main stack": `|7, 8|;`
5. Gli ultimi due oggetti inseriti vengono eliminati dallo stack e comparati per confrontarne l'uguaglianza - di conseguenza risultano non uguali, e quindi il booleano *FALSE(0)* viene inserito nello stack: `|0|`.

Alla fine dell'esecuzione del programma, l'oggetto in cima al main stack è il valore del risultato del programma.

Locking e Unlocking Scripts: Il trasferimento del valore di Bitcoin viene effettuato dagli inputs e dagli outputs, come descritto nella sezione precedente. Questa sezione invece descrive il meccanismo di autorizzazione delle transazioni Bitcoin. L'output di una transazione contiene un programma chiamato locking script. Tale programma definisce come e se l'output può essere speso. In modo simile, ogni input contiene un programma chiamato unlocking script. L'unlocking script non è altro che la prova della corrispondenza con il locking script per permettere all'output di essere speso. I locking e unlocking scripts, assieme, implementano il meccanismo di autorizzazione

del trasferimento di valore. I nodi completi Bitcoin eseguono la seguente procedura per verificare l'autorizzazione a "spendere" (cioè, validare ed inserire nella blockchain) una transazione.

- L'input *I*, che corrisponde ad un output *O*, viene selezionato dalla transazione;
- L'unlocking script dell'input *I* viene eseguito nello stack *S*;
- Se l'esecuzione dell'unlocking script non presenta alcun errore, il locking script dell'ouput *O* viene eseguito con il restante contenuto dello stack *S*;
- Se il locking script termina senza errori e il valore di risultato dello stack *S* è *TRUE*, l'input *I* viene autorizzato a spendere l'output *O*;
- In altri casi (se ci sono degli errori nell'esecuzione di uno dei due scripts o il valore di risultato non è *TRUE*) la procedura termina e la transazione non è considerata valida;
- Un altro input con il suo corrispondente output viene selezionato e il processo si ripete finché la transazione non viene considerata valida Il locking script è chiamato anche *Pubkey Script* (o *ScriptPubKey*) nelle ricerche accademiche su Bitcoin [52]; similarmente, l'unlocking script è chiamato anche *Signature Script* (*ScriptSig*) [53]. La ragione di questi nomi risiede nella storia del Bitcoin e nella sua natura crittografica: nelle prime versioni dei software Bitcoin gli output includevano anche la chiave pubblica (*public key*), mentre l'input corrispondente doveva dimostrare il possesso della chiave privata includendo una firma (*signature*). Il meccanismo di autorizzazione verificava la chiave pubblica e la firma e permetteva di "sbloccare" l'output, se la firma era corretta. Questo schema d autorizzazione è diventato obsoleto nel 2016 , ma i nomi sono ancora in uso. Possiamo definire il bilancio dell'utente come la somma degli output non spesi e che possono essere spesi dall'utente; difatti, i wallet per Bitcoin calcolano il bilancio con questa metodologia.

In modo simile, la somma di tutti gli output non spesi di Bitcoin creati fino ad ora equivale al valore totale dei bitcoin in circolazione.

Transazioni e scripts standard: Bitcoin Core definisce cinque tipologie di transazioni che vengono chiamate "standard" [54]. Una transazione è standard se risulta positiva al test incluso nella funzione `IsStandard`. Le transazioni standard sono comunemente usate all'interno della rete Bitcoin e con lo sviluppo potrebbero essere modificate e aggiornate (esempio di proposta di modifica `https://gist.github.com/gavinandresen/88be40c141bc67acb247`). Le transazioni standard permettono il funzionamento degli scripts standard (cioè degli schemi di autorizzazione), riportati qui sotto. Una transazione standard può contenere multipli outputs con differenti scripts standard. Tuttavia, se una transazione contiene almeno uno script non-standard, la transazione viene catalogata come non-standard.

1. Pay-To-Public-Key (P2PK)

```
LS: <public_key> OP_CHECKSIG
ULS: <signature>
```

   Questo schema di autorizzazione è stato trattato anche nella sezione precedente. Come prima operazione, l'unlocking script (ULS) viene eseguito e la firma (signature) viene aggiunta al main stack. Il locking script viene successivamente eseguito, la chiave pubblica viene inserita nel main stack e la funzione `OP_CHECKSIG` viene chiamata. Tale funzione prende come argomenti gli elementi presenti nello stack e controlla se la firma creata dalla chiave privata corrisponde alla chiave pubblica. Se e solo se la firma è corretta, l'input è autorizzato a spendere l'output. In altre parole, solamente il proprietario della chiave pubblica archiviata nell'output può spendere quest'ultimo.

2. Pay-to-Public-Key-Hash (P2PKH)

```
LS: OP_DUP OP_HASH160 <public_key_hash> OP_EQUAL OP_CHECKSIG
ULS: <signature> <public_key>
```

Il *Pay-to-Public-Key-Hash* è funzionalmente simile al *Pay-To-Public-Key*; come prima cosa, l'unlocking script viene eseguito e sia la firma che la corrispondente chiave pubblica contenute nell'input vengono caricate nello stack. Viene poi eseguita la funzione `OP_DUP` duplica la chiave pubblica mentre la funzione `OP_HASH160` crea l'hash della chiave pubblica duplicata. Successivamente l'hash della chiave pubblica contenuta nell'output viene inserita nello stack, dove la funzione `OP_EQUAL` verifica che l'hash proveniente dall'output e l'hash della chiave pubblica inclusa nell'input siano uguali. Le restanti operazioni incluse nello schema di autorizzazione sono le stesse del *Pay-To-Public-Key*, dove la funzione `OP_CHECKSIG` controlla che la firma corrisponda con la chiave pubblica. Questo schema è di uso comune e viene anche chiamato "*Pay-to-Address*". Il `<public-key-hash>` contenuto nell'output non è altro che l'indirizzo Bitcoin senza la codifica `BASE58CHECK` e senza il `version_byte` (il primo carattere dell'indirizzo Bitcoin). Uno dei vantaggi principali del Pay-To-Public-Key-Hash è l'assenza di una chiave pubblica non criptata nell'output; questo può essere una fonte di rischio in quanto è possibile, avvalendosi teoricamente di un quantum computer e utilizzando degli algoritmi specifici (come l'algoritmo di Shor o di Grover [55]), gli hacker potrebbero utilizzare i bitcoins non ancora spesi provenienti dall'output.

3. Multi-Signature (Multisig)

```
LS: <M> <public_key_1> ... <public_key_N>
<N> OP_CHECKMULTISIG
ULS: <signature_i> ... <signature_j>
```

Tale schema pone una condizione nella quale N chiavi pubbliche vengono registrate nello script e almeno *M* di queste devono fornire una firma valida. Qualsiasi combinazione di M firme è accettata. Questo schema è conosciuto anche con il nome "M-of-N scheme", dove *N* è il numero totale di chiavi pubbliche e *M* è il numero minimo di firme richieste per rendere valida la transazione. L'output del Multi-Signature può implementare un metodo di autorizzazione basato su *N* fattori, richiedendo quindi *N* devices elettronici, come computer o smartphone, per autorizzare la transazione di bitcoins.

4. Pay-to-Script-Hash (P2SH)

```
LS: OP_HASH160 <redeem_script_hash> OP_EQUAL
ULS: <inner_unlocking_script> "<redeem_script>"
```

Il Pay-to-Script-Hash è un cosiddetto schema "wrapper", cioè può "contenere" altri schemi di autorizzazione. Il Multisig o altre tipologie di schemi possono essere trasformati in P2SH in modo tale che l'hash dell'unlocking script originale venga piazzato fra le funzioni `OP_HASH160` e `OP_EQUAL` (è chiamato "*redeem script hash*"). L'unlocking script e il locking script originali vengono inseriti nel P2SH unlocking script rispettivamente con il nome *inner unlocking script* e *redeem script**; quest'ultimo viene serializzato, cioè viene convertito in una stringa, in forma binaria o attraverso una codifica testuale, per poi essere deserializzato successivamente. Tale processo permette di trasmettere l'intero stato dell'oggetto serializzato in modo che possa essere successivamente ricreato nello stesso identico stato dal processo inverso, la deserializzazione. Oltre a ciò, l'esecuzione dello script differisce leggermente dagli altri schemi di autorizzazione; per prima cosa, l'unlocking script viene eseguito, eseguendo di conseguenza l'*inner unlocking script* e inserendo nello stack il *redeem script* serializzato. Poi la funzione `OP_HASH160` crea l'hash del redeem script già presente nello stack e viene inserito anche l'hash del redeem script

contenuto nel locking script. Infine l'`OP_EQUAL` compara l'hash appena calcolato con l'hash inserito nello stack. Da quest'ultimo passaggio in poi, il processo d'esecuzione inizia a differire: il redeem script serializzato viene automaticamente rimosso dallo stack e viene deserializzato. A questo punto dell'esecuzione, lo stack contiene ancora i risultati provenienti dall'inner unlocking script; il redeem script deserializzato viene poi eseguito utilizzando lo stack come se fosse un unlocking script. Lo schema P2SH ha vari vantaggi: considerando due utenti, un ricevente ed un mittente, e quest'ultimo crea una transazione (utilizzando gli input di una transazione passata e creando degli outputs nella nuova transazione), entrambi gli outputs devono contenere il locking script.

Altri script:

- se il ricevente vuole ottenere la proprietà dei bitcoins con lo script *Pay-to-Public-Key* allora la chiave pubblica del ricevente dev'essere conosciuta in anticipo dal mittente;
- se il ricevente vuole ottenere la proprietà dei bitcoins con lo script *Pay-to-Public-Key-Hash* allora l'hash della chiave pubblica del ricevente dev'essere conosciuta in anticipo dal mittente;
- se il ricevente vuole ottenere la proprietà dei bitcoins utilizzando lo script *Multi-Signature* allora la lista delle N chiavi pubbliche dev'essere conosciuta in anticipo dal mittente;
- se il ricevente vuole ottenere la proprietà dei bitcoins con uno script *nonstandard* allora la parte del locking script dev'essere conosciuta in anticipo dal mittente.

Con lo script Pay-to-Script-Hash, il mandante deve essere a conoscenza solamente dell'hash del redeem script, il quale ha sempre una lunghezza fissa e può essere considerato una richiesta di pagamento universale, in quanto l'hash del redeem

script può essere qualsiasi standard locking script. In questo modo, inoltre, il mandante della transazione o i nodi del network Bitcoin non hanno bisogno di sapere come verrà utilizzato l'output in futuro. Con il Pay-to-Script-Hash, l'indirizzo Bitcoin non incorpora solamente il concetto di mandare denaro ad un ricevente, piuttosto rappresenta il concetto di "denaro programmabile" dove il ricevente può decidere cosa farne del denaro.

5. Data Output (`OP_RETURN`)

```
LS: OP_RETURN <data>
```

I locking script e le transazioni che contengono l'operatore `OP_RETURN` non sono spendibili. La funzione dell'`OP_RETURN` è quella di bloccare l'esecuzione di uno script e rendere le transazioni non valide. Tuttavia, un locking script con `OP_RETURN` può contenere dati arbitrari di dimensione fino a 40 bytes; questa è sufficiente a contenere l'hash di algoritmi come lo `SHA256`. Un servizio notarile (www.proofofexistence.com) online permette infatti di utilizzare outputs con operatore `OP_RETURN` e 40 bytes di spazio per memorizzare l'hash dei file. Questo permette di provare l'esistenza del file in un certo istante temporale, e cioè quando il blocco contenente la transazione è stato incluso nella blockchain (verrà trattato più approfonditamente nel seguente paragrafo). Infine, non è richiesta alcun servizio online centralizzato per archiviare l'hash del file; tutte le transazioni, infatti, vengono propagate direttamente nella rete peer-to-peer Bitcoin.

A.4 Blockchain

Le transazioni contenenti nuovi bitcoins vengono ricevuti dai nodi dei miners e poi memorizzate in strutture di dati chiamate blocchi. I miners inviano i blocchi che contengono le transazioni agli altri nodi appartenenti alla rete. Da qui il nome blockchain, cioè

una struttura che consiste in una serie di blocchi concatenati l'un l'altro. La blockchain contiene al suo interno tutti gli outputs e gli inputs creati fino ad una determinata data; quest'ultima è la data in cui il consenso distribuito dev'essere raggiunto.

La struttura dei blocchi

Come mostrato nella Tabella A.5, un blocco contiene le informazioni relative alla sua dimensione, alla testata del blocco contenente alcuni metadati e una lista delle transazioni.

Tabella A.5: Struttura del blocco

Dimensione	Campo
4 bytes	Dimensione del blocco (Block Size)
80 bytes	Testata del blocco (Block Header)
1-9 bytes	Numero della transazione (Transaction Counter)
Variabile	Transazione (outputs e inputs)

La testata di un blocco contiene dei metadati necessari per il processo di data consistency e alcune informazioni sono usate per il mining.

Tabella A.6: Struttura della testata del blocco

Dimensione	Campo
4 bytes	Versione
32 bytes	Hash della testata del blocco precedente
32 bytes	Merkle Root
4 bytes	Timestamp
4 bytes	Difficulty Target
4 bytes	Nonce

Un blocco è univocamente identificato con un digest crittografico (un doppio `SHA256`) della propria testata. Come mostrato nella Tabella A.6, l'header del blocco contiene l'hash del header del blocco precedente, in modo da creare una parentela di riferimento

con altri blocchi. Il primo blocco, chiamato "Genesis Block", è stato creato manualmente e non ha alcun riferimento ad un blocco precedente (il campo infatti contiene 0-bytes).

Merkle tree

Un albero Merkle (Merkle Tree) è una struttura di dati basata su hash. Una struttura dati basata su hash assegna i dati a una chiave: ciò può essere paragonato a quando viene effettuata una chiamata rapida nel telefono, dove ogni tasto ha assegnato un determinato numero di telefono (la chiave è il tasto, mentre l'informazione associata è il numero di telefono). Gli alberi Merkle vengono comunemente usati per verificare l'integrità dei dati. E' una struttura molto efficiente in quanto vengono utilizzati degli hash anziché un file completo di informazioni. Gli alberi Merkle tipicamente hanno un fattore a due rami (branches), il che significa che ogni nodo può avere due figli (child). Anche in questo caso, gli alberi di Merkle sono generalmente implementati come alberi binari (on/off o 1/0), ma ancora una volta potrebbero essere creati come alberi con n figli per nodo.

Catena di blocchi (*chain of blocks*)

Ogni blocco è associato agli altri blocchi facendo riferimento ad un blocco parente; se qualsiasi informazione contenuta nel header di un blocco viene modificata, allora l'header del suo blocco figlio conterrà un hash non valido. Inoltre è possibile rilevare la modifica di una transazione: l'header di un blocco contiene la cosiddetta Merkle root, cioè la radice di una struttura ad albero Merkle dove le transazioni rappresentano le foglie. La Merkle root contenuta nel header del blocco ha il ruolo di impronta delle transazioni nello stesso blocco.

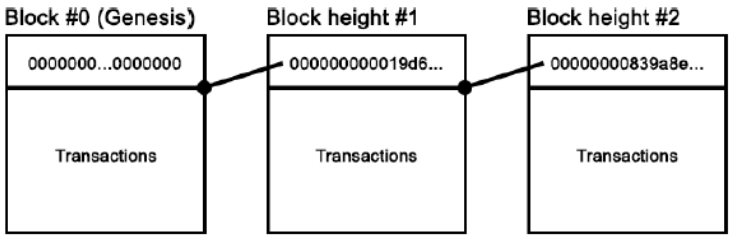

Figura A.2: Struttura di collegamento dei blocchi

Se una transazione presente nella blockchain viene alterata, rimossa o aggiunta successivamente al blocco è facilmente identificabile tramite la Merkle root contenuta nel blocco. Quando cambia la Merkle root, l'hash del header in cui è contenuta è differente rispetto al hash dell header del blocco precedente contenuto nel blocco figlio. Questo poi porta ad invalidare la correttezza del successivo hash del header del blocco nipote. In pratica, un cambiamento in anche solo una transazione intacca la correttezza di tutti i blocchi discendenti. La Figura A.2 raffigura i primi tre blocchi di Bitcoin. Non è una coincidenza che gli hash del primo e del secondo blocco inizino con una serie di zero. Il motivo di ciò verrà spiegato nel seguente paragrafo.

A.5 Consenso distribuito (*distributed consensus*)

Nelle architetture centralizzate, come un'architettura server-client, gli utenti si fidano del fatto che il server (cioè l'autorità centrale) gli fornisca informazioni affidabili. Per proteggere la validità dei dati, possono essere implementate una serie di misure addizionali, ad esempio degli integrity checks o dei processi di audit sui dati. Tuttavia, questo crea una situazione in cui l'autorità centrale non è sorvegliata e in cui bisogna riporre completa fiducia. In un architettura decentralizzata, come per Bitcoin, i nodi non ripongono alcuna fiducia neghi altri nodi e i dati che ricevono vengono sempre considerati corrotti o fraudolenti. Invece di una serie di regole definite dalle autorità, Bitcoin

utilizza le *consensus rules*, cioè delle regole che permettono di raggiungere un consenso generale sulla validità delle informazioni in analisi [56]. Se un nodo non aderisce a queste regole non è autorizzato a partecipare al network e le informazioni riguardanti la blockchain in lui contenute non sono consistenti comparate a quelle contenute negli altri nodi. Per esempio, se un nodo crea una più alta ricompensa per il mining rispetto a quella consentita, la consensus rule viene cambiata e i bitcoins (il coinbase input) non possono essere archiviati nella blockchain. Tuttavia, i nodi che mantengono questa modifica, accettando quindi una ricompensa per il mining più alta, hanno la possibilità di formare una blockchain alternativa e incompatibile (in quanto proveniente da un hark-fork, termine che in informatica indica il momento in cui un progetto si divide in due rami distinti e con funzioni diverse) con la precedente blockchain. Nella letteratura riguardante Bitcoin, la parola "consenso" è intesa in modo generale e spesso riferita al concetto di consenso emergente, come riportato anche da Andreas M. Antonopoulos in "Mastering Bitcoin":

> "Satoshi Nakamoto's main invention is the decentralized mechanism for emergent consensus. Emergent, because consensus is not achieved explicitly- there is no election or fixed moment when consensus occurs. Instead, consensus is an emergent artifact of the asynchronous interaction of thousands of independent nodes, all following simple rules. All the properties of bitcoin, including currency, transactions, payments, and the security model that does not depend on central authority or trust, derive from this invention." [46]

Verifica delle transazioni

Nella Sezione A.3 è stato brevemente descritto il fatto che i nodi Bitcoin seguono un protocollo di regole per verificare le nuove transazioni che devono essere inserite nella rete. Le regole, come detto in precedenza, hanno l'obiettivo di ottenere un consenso

distribuito, cioè quando la maggior parte dei nodi è in accordo con la correttezza e la validità delle informazioni da inserire nella blockchain. Tuttavia, sono incluse anche ulteriori regole riguardanti, ad esempio, la sicurezza[7]; queste regole inoltre sono in continua evoluzione e c'è la possibilità che ne vengano aggiunte in futuro.
Viene riportata qui di seguito una lista delle regole più importanti per quanto riguarda la rete Bitcoin:

- La transazione dev'essere corretta in tutti i suoi parametri;
- La dimensione del blocco in bytes dev'essere sempre minore o uguale alla dimensione massima del blocco stabilita nella blockchain (denominata con la dicitura `MAX_BLOCK_SIZE`);
- La transazione non dev'essere già inclusa nella blockchain;
- La somma dei valori ad input dev'essere maggiore o uguale alla somma dei valori in output.

A.6 Mining

I nodi appartenenti alla rete Bitcoin collezionano, verificano e propagano in continuazione le nuove transazioni che vengono create. Le transazioni vengono raccolte localmente nella *transaction pool* (chiamata anche *memory pool*) dei nodi con funzione di router [46]. I nodi mining (detti *miners*) sono coinvolti in un processo chiamato mining, il quale, una volta consensualmente riconosciuto come valido da tutti i nodi, permette di "costruire" blocco per blocco la blockchain. Si prenda in considerazione, a titolo esemplificativo, un nodo mining che propaga una nuova transazione ad un blocco "candidato": tale blocco viene temporaneamente archiviato nella memoria del nodo

[7]Sono presenti regole per prevenire i cosi detti "*Denial of Service Attacks*", che nel campo della sicurezza informatica indica un malfunzionamento dovuto ad un attacco informatico in cui si fanno esaurire deliberatamente le risorse di un sistema informatico che fornisce un servizio ai client, ad esempio un sito web su un web server, fino a renderlo non più in grado di erogare il servizio ai client richiedenti.

miner. L'height number, cioè il numero totale di blocchi connessi al blocco di genesi della blockchain, del blocco candidato è $N + 1$ dove N è il numero dell'ultimo blocco conosciuto dai miners incluso nella blockchain. In altre parole, il blocco candidato è un potenziale figlio del blocco con height number uguale a N. Inoltre, si supponga che, mentre il blocco candidato è archiviato temporaneamente e il processo di mining è in atto, un nuovo blocco venga inviato da un altro miner. Questo viene verificato e archiviato dal miner come ultimo blocco ricevuto nella blockchain. Il miner poi non fa altro che filtrare i dati ricevuti con quelli già presenti e li include nella transaction pool. Infine un nuovo blocco candidato viene creato, il quale non includerà nessuna delle precedenti transazioni.

Selezione delle transazioni: Alcune transazioni hanno una probabilità più alta di essere incluse nel blocco rispetto ad altre. La funzione [46] per il calcolo della priorità delle transazioni (*transaction priority*) è:

```
transaction_priority = SUM(value_of_input*input_age)/transaction size
```

Il *value_of_input* è il valore dei bitcoins in unita satoshi (1 bitcoin = $10e-8satoshi$). L'*input_age* è il numero di blocchi ricevuti da quando la transazione è stata archiviata nella blockchain. La *transaction_size* invece è la dimensione della transazione in bytes.

```
high_priority>(100000000*144)/250=57600000
```

I primi 50 kilobytes dello spazio disponibile alle transazioni viene allocato per transazioni con alta priorità (priorità maggiore di 57600000 [57]. L'alta priorità, high priority, è una soglia che rappresenta una transazione con un bitcoin (100000000 satoshi), "invecchiato" 144 blocchi (approssimativamente 24h) e incluso in una transazione da 250 bytes. La logica utilizzata è che se una transazione contiene un valore ingente o è in attesa datroppo tempo per essere inclusa, allora verrà prioritarizzata. Il miner poi riempirà lo

spazio rimanente della `MAX_BLOCK_SIZE` con altre transazioni. Generalmente i miners includono transazioni con commissioni miners più alte per massimizzare il loro profitto.

Metodo di calcolo della ricompensa per i Miners: I miners includono nel blocco candidato una transazione (detta coinbase) addizionale. Le transazioni coinbase contengono di solito un output con scripts Pay-to-Public-Key O Pay-to-Public-Key-Hash con l'indirizzo del nodo miner. Questa commissione è calcolata in base al blocco con *height N* più alta incluso nella blockchain. L'height *N* può essere interpretata come una sorta di orologio della rete Bitcoin, che difatti può essere approssimato al tempo fisico [58].

```
coinbase_reward = calculateReward(N)
```

La commissione originalmente era di 50 bitcoins, tuttavia ogni 210000 blocchi (approssimativamente 4 anni), questa viene ridotta del 50%. Questo evento è conosciuto con il nome block reward halving (letteralmente, dimezzamento della commissione). Con questa offerta di bitcoins controllata, non ci saranno più di 21 milioni di bitcoins in esistenza [59]. La Formula A.6 mostra la formula per il calcolo della commissione per i miners. Questa fee verrà ridotta quando la block reward sarà diminuita.

```
miner_fee = SUM(included_inputs)-SUM(included_outputs)
```

Infine, la reward totale che andrà al miner è la somma della *miner fee* e la *coinbase reward*

```
total_reward = miner_fee + coinbase_reward
```

Finalizzazione del blocco: Quando un miner inserisce una transazione coinbase e un subset delle transazioni presenti nel blocco candidato, la sua struttura dev'essere aggiornata.

Dimensione	Campo
4 bytes	Versione
32 bytes	Hash della testata del blocco precedente
32 bytes	Merkle Root
4 bytes	Timestamp
4 bytes	Difficulty Target
4 bytes	Nonce

La versione del blocco è settata sulla versione corrente. L'hash crittografico della testata del blocco con *height N* viene calcolato e poi archiviato nella testata del blocco candidato. Sia Merkle root (il digest crittografico delle transazioni che vengono incluse) che il timestamp vengono aggiornati. Il *difficulty target* è inizializzato ad un determinato valore che viene misurato ogni 2016 blocchi; è un parametro importante per la *proof-of-work*, un sistema che verrà descritto nel prossimo paragrafo. Infine il *nonce* ha come valore zero. La selezione delle transazioni, il calcolo della commissione per i miners e la finalizzazione del blocco sono procedure attuate non da un solo nodo miner ma piuttosto da tutti i nodi in modo concorrenziale. Tuttavia, la blockchain è costruita blocco-per-blocco e tutti i nuovi nodi hanno bisogno di essere serializzati. Il sistema proof-of-work è usato sia per settare l'ordine dei blocchi che per prevenire problemi come la congestione della rete.

Proof-of-work: Questo processo può essere paragonato ad un'autorità "onesta" che sceglie in modo casuale un miner fra tutti quelli disponibili ogni 10 minuti. Il miner "prescelto" può propagare il suo blocco prescelto agli altri nodi della rete. Questo approccio dovrebbe rimuovere la congestione della rete; Bitcoin utilizza il sistema proof-of-work [60] con algoritmo *Hashcash* [61]. L'algoritmo della proof-of-work usato dai miners di Bitcoin può essere espresso in *pseudo code*. La funzione sottostante usa come parametri la testata del blocco e un target. L'output che restituisce è una testata del blocco modificata che potenzialmente può rappresentare quella del prossimo blocco

nella blockchain. Il processo è iterativo, e la funzione cambia continuamente il campo *nonce* in un blocco fino a che un corretto hash crittografico viene trovato e così il valore del *nonce* viene inserito nella testata del blocco).

```
block_header proofOfWork (block_header, target) {
     while (true) {
          hash = cryptographic_hash (block);
          if (hashReachedTarget(hash, target)) {
               return block_header;
          }
          bloc_header.nonce++;
     }
}
```

E una possibile rappresentazione della funzione `hashReachedTarget` è:

```
boolean hashReachedTarget (hash, target) {
     return hash == target;
}
```

Bitcoin utilizza una doppia funzione `SHA256`, `SHA256(SHA256())`, come funzione per creare l'hash crittografico; la funzione restituisce 256 bit di digest[8]. Anche il target ha una dimensione di 256 bit [46]. Un hash raggiunge il target se e solo se l'hash è minore del target. Il target parametrizza il numero medio dei calcoli per trovare l'hash corretto. Ogni miner Bitcoin ricalcola indipendentemente il target ottimale ogni 2016 blocchi; il risultato è chiamato difficulty, la difficoltà, ed è codificata e rappresentata in un formato differente rispetto al target. La difficoltà rappresenta il target per i successivi 2016 blocchi dal suo calcolo.

[8]Il digest è l'output che viene prodotto da una funzione di hash

```
difficulty= old_difficulty*(minutes_between_blocks(N, N-2016)/2016*10)
```

L'equazione per la difficoltà A.6 misura i minuti fra l'ultimo 2016esimo blocco e aggiusta la vecchia difficoltà per rappresentare un target di 10 minuti in media per trovare il blocco corretto. L'aggiustamento è impostato intenzionalmente per produrre una media mobile [62]. Se ogni nodo miner aggiorna l'hardware per essere due volte più veloce nel momento in cui la blockchain è a 1008 blocchi dopo l'ultima ricomputazione, la nuova difficoltà sarà solo 50% più alta.

A.7 Validazione dei blocchi indipendente

Quando un miner trova il nonce corretto, propaga immediatamente il blocco candidato agli altri nodi. Ogni nodo esegue un controllo [63] per validare il nuovo blocco ricevuto. Se viene validato allora viene "incatenato" nella copia locale della blockchain e propagata a tutti gli altri nodi del network. Una breve lista dei controlli inclusi può essere:

- il blocco dev'essere sintatticamente corretto;
- l'hash della testata del blocco dev'essere minore del target;
- la prima transazione è una transazione coinbase;
- soltanto una transazione coinbase dev'essere inclusa;
- tutte le transazioni incluse devono essere valide.

A.8 Implicazioni per la sicurezza

Se la blockchain ri-converge in differenti catene con potenziali differenti transazioni, i miners "onesti" posso includere le transazioni della catena invalida nella transaction

pool. Tuttavia, possono esistere anche miners "malevoli" nel sistema. Le transazioni Bitcoin utilizzano il concetto di confirmation [64]: una transazione inclusa nella blockchain in un blocco con N successori ha una confirmation $N+1$. Ad esempio, due gruppi di nodi miners dove ogni gruppo ha esattamente il 50% dell'hashing power totale (la potenza di elaborazione della rete), il gruppo A e il gruppo B. In una prospettiva di lungo periodo, il 50% dei blocchi dovrebbe venir minato dal gruppo A e il 50% dal gruppo B; nonostante ciò, se il gruppo A accumula più del 50% dell'hashing power, produrrebbe eventualmente più blocchi del gruppo B.

Majority attack

Si consideri un individuo con intenzioni malevoli che controlla un gruppo di miners con più del 50% dell'hashing power totale. Si definisca inoltre un blocco X conosciuto da tutti i nodi del network; il gruppo di miners controllati inizia il processo di minting ("coniare") di una catena segreta dal blocco X - una catena segreta non viene pubblicata agli altri miners. Supponendo che la difficoltà del mining non è cambiata dal blocco X, la catena segreta risulterà quella con la catena più lunga e con un grado di difficoltà accumulato più alto. Finché la catena segreta continua ad avere la difficoltà cumulata più alta, può essere propagata agli altri miners blocco per blocco nello stesso momento. Tutti i miners setteranno la catena segreta come la nuova main chain, rimpiazzando effettivamente tutti i blocchi dal blocco X. Questo tipo di attacco è conosciuto con il nome di "majority attack", "50% attack" o "51% attack". Questo attacco appartiene alla categoria dei *brute force attacks* [65]; con il majority attack, un aggressore può [66]:

- Modificare tutte le transazioni incluse dopo il blocco X;
- Modificare le transazioni incluse prima o nel blocco X;
- Convertire le sue stesse transazioni, rimpiazzandole con transazioni già utilizzate nella secret chain, facendo emergere il problema del double-spending;

- Scegliere quali transazioni includere permanentemente nella blockchain Tuttavia, un aggressore non può [66]:
  - Inviare bitcoins che non gli appartengono;
  - Creare denaro dal nulla;
  - Far accettare agli altri nodi le sue consensus rules;
  - Convertire le transazioni di altre persone senza la loro cooperazione ;
  - Bloccare transazioni non confermate dalla rete.

Considerando un miner con il 10% del hashing power totale, questo ha una probabilità del 20% di eseguire con successo un tentativo di double-spending su una transazione con una confirmation [65]. Lo stesso miner ha una probabilità di successo del 5.6% con una transazione con due confirmation e lo 0,0059% di probabilità di successo con una transazione con sei confirmation. Nonostante ciò, il mining di una catena segreta che non verrà inclusa nella blockchain ha un costo per l'operatore del mining in termini di elettricità; inoltre, gli fa perdere sia la ricompensa del blocco che le commissioni sulle transazioni nella catena segreta non inclusa. Infatti, gli operatori di mining hanno un incentivo economico a non attaccare la rete Bitcoin, in quanto hanno investito nell'hardware per effettuare il mining e si aspetta un ritorno da questo investimento o comunque dei futuri profitti. Il mining decentralizzato insieme agli incentivi economici mantengono, nella pratica, immutata la blockchain. Dalla prospettiva di chi deve ricevere i bitcoins, essi devono stimare un numero di confirmations C tale che C sia alto abbastanza per considerare una transazione ricevuta immutabile. Per esempio, la probabilità di successo di un *doublespending attack* che riesca a convertire delle transazioni è minore se $C = 6$ piuttosto che $C = 1$ [65]. Il numero di confirmations sufficiente a rendere una transazione "sicura" può essere deciso da chi deve ricevere i bitcoins in base al rischio di fronde a cui è sottoposto (molti exchange di crytpovalute, soprattutto quelli

centralizzati, richiedono di solito almeno 6 confirmations per permettere agli utenti di utilizzare i fondi trasferiti nell'exchange).

Appendice B

Piattaforma Ethereum e Smart Contracts

B.1 Contratti

Un contratto è un accordo volontario fra due o più parti, il quale può essere impugnabile in quanto accordo legale vincolante [67]. I contratti vengono solitamente stabiliti da entrambe le parti e contengono un testo negoziabile che dev'essere comprensibile e chiaro. Un intermediario (ad esempio, un notaio) può firmare il contratto per proteggerne l'autenticità e le autorità hanno la possibilità di mettere forzatamente in atto quanto riportato e definito all'interno del contratto. Considerando invece un contratto digitalizzato, le parti contraenti possono essere rappresentate da un numero identificativo e il contenuto del contratto può essere archiviato sotto forma di codice informatico, utilizzando parametri condizionali, regole e procedure. Per evitare confusione nella terminologia dei contratti digitalizzati, verranno analizzate due differenti [68], anche se correlati, concetti di contratto.

B.1.1 Contratto ricardiano

Il *ricardian contract* è un template per contratti che ha la caratteristica di poter essere compreso da umani e da computers ed è considerato un pattern da utilizzare nell'architettura di software che interagiscono con i contratti [69]. Questa tipologia di contratto è stata inventata da Ian Grigg nel 1995 con lo scopo di portare ed emettere strumenti finanziari nel Web[70]:

> "A Ricardian Contract can be defined as a single document that is a) a contract offered by an issuer to holders, b) for a valuable right held by holders, and managed by the issuer, c) easily readable by people (like a contract on paper), d) readable by programs (parsable like a database), e) digitally signed, f) carries the keys and server information, and g) allied with a unique and secure identifier.[71]"

B.2 Smart contract

Uno *smart contract* è un protocollo che facilita, verifica o fa rispettare quanto contenuto in un contratto [72]. Questo termine è stato definito da Nick Szabo fra il 1994 e il 1997, il quale, nei suoi lavori, ha definito una serie di possibili applicazioni di esso:

> "A smart contract is a computerized transaction protocol that executes the terms of a contract. The general objectives of smart contract design are to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement), minimize exceptions both malicious and accidental, and minimize the need for trusted intermediaries. Related economic goals include lowering fraud loss, arbitration and enforcement costs, and other transaction costs." [73]

B.2.1 Esempi di contratto ricardiano

Generalmente i contratti richiedono una buona condotta da entrambe le parti coinvolte in esso per essere considerato valido. In modo simile, la buona condotta è richiesta per implementare quanto definito dal contratto. Le autorità riconosciute dalle parti possono forzatamente sostituirsi alla buona condotta. Tuttavia, il coinvolgimento delle autorità comporta un ulteriore costo per le parti ed esiste la possibilità che esse siano disoneste o corruttibili. I contratti ricardiani possono essere formalizzati e digitalizzati in codice informatico tramite l'utilizzo di linguaggi di programmazione e possono essere memorizzati in un personal computer, in un server o in una piattaforma cloud. Tale codice può essere utilizzato per valutare le informazioni contenute nell'input in modo che rispettino le regole definite dal contratto e produrre un output coerente ad esse. Il network Bitcoin può essere utilizzato come una piattaforma per archiviare ed eseguire i contratti formalizzati. Tuttavia, in contrasto con gli altri sistemi o piattaforme centralizzate, la piattaforma Bitcoin non è controllata da una singola autorità o da una catena di autorità.

Escrow (garanzia) e mediazione delle dispute

Si consideri un compratore, un venditore e un mediatore. In questo esempio, il venditore vende wallets hardware. Il compratore, però, vive a 500 km dal venditore e entrambi hanno una bassa reputazione. Il mediatore invece ha un alta reputazione (in questo caso, la reputazione viene considerata come il grado di affidabilità di un individuo). Un possibile contratto per garantire la compravendita può essere:

- Il compratore alloca D denaro in una transazione
- Il denaro allocato non può essere utilizzato
- Se il compratore e il venditore sono d'accordo (cioè quando la compravendita avviene con successo), il denaro viene accreditato al venditore

- Se il compratore e il mediatore sono d'accordo (cioè quando la compravendita fallisce e il mediatore si schiera dalla parte del compratore), il denaro viene restituito al compratore

- Se il venditore e il mediatore sono d'accordo (cioè quando il cliente è in torto), il denaro viene accreditato al venditore

In tutti i casi in cui l'accredito del denaro avviene con successo, il mediatore ne guadagna in affidabilità; se tale reputazione è abbastanza alta, il mediatore è incentivato a risolvere le potenziali dispute. Il contratto può essere inoltre eseguito in un secondo momento. Per esempio, una parte del denaro può essere accreditata al mediatore nel caso in cui esso risolva una disputa; questo aumenta gli incentivi a breve termine per il mediatore. Inoltre, se la disputa non viene risolta, il denaro allocato può essere diviso automaticamente fra il compratore e il venditore dopo G giorni. Il contratto che può essere implementato in Bitcoin come uno smart contract può avere uno schema di autorizzazione Multi-Signature di 2-of-3 [74].

B.3 Ethereum

Ethereum è una piattaforma decentralizzata che permette di eseguire gli smart contracts. Similmente a Bitcoin, Ethereum utilizza dei tokens interni chiamati *ethers* con un'unità atomica chiamata *wei* (1 ether = 10^{18} wei). Il modello di emissione degli ethers è disinflazionistico[1] [75] con un valore massimo, non costante, per ogni blocco. Esistono due definizioni comuni per il termine "inflazione": la prima si riferisce ai prezzi, mentre la seconda è relativa alla quantità totale di denaro all'interno di un sistema (detta la base monetaria o offerta). Bisogna quindi distinguere fra "inflazione del prezzo", che indica l'aumento del prezzo generale dei beni o servizi in un'economia,

[1] In economia la disinflazione è una riduzione dell'inflazione a seguito di un decremento del tasso di inflazione, in questo caso inflazione monetaria. Di conseguenza la crescita dei prezzi rallenta: per esempio, se il tasso di inflazione passa dal 6% al 2% la disinflazione è pari al 4% e i prezzi dei beni crescono di meno.

e "inflazione monetaria", intesa come la crescita dell'offerta di moneta all'interno di un'economia dovuta ad un determinato meccanismo di emissione. Spesso, ma non sempre, l'inflazione monetaria è causata dall'inflazione del prezzo. Anche se l'emissione di *ethers* rimane costante, il tasso di crescita della base monetaria non è costante. Questo tasso di inflazione monetaria diminuisce ogni anno facendo di Ethereum una valuta deflazionistica, in termini di base monetaria. La blockchain di Ethereum, nel 2019, è costituita da nodi mining che utilizzano l'algoritmo *Ethash* come proof-of-work. Il tempo medio per la formazione di un blocco è 15 secondi, mentre la ricompensa per i miners che creano un blocco candidato è di 5 ethers [75].

B.3.1 Concetti

Anche se sia Bitcoin che Ethereum sono un sistema peer-to-peer e utilizzano dei concetti comuni come il mining, una blockchain distribuita, sono internamente differenti e le loro implementazioni differiscono. Infatti, queste differenze rendono Ethereum una piattaforma molto più adatta alla programmazione di smart contracts con una logica complessa.

Accounts

Bitcoin utilizza un paradigma per le transazioni utilizzando gli output come rappresentazione del valore contenuto. Ethereum invece usa un paradigma alternativo che si basa sui contratti per definire uno schema di autenticazione avanzato e altre logiche di programmazione. Esistono due tipologie astratte di account:

- **Account con proprietà esterna** (*externally owned account*): gli account di questo tipo (mostrati nella tabella sottostante) sono controllati da una chiave privata e identificati da un indirizzo derivato dalla corrispondente chiave pubblica con una funzione di hash crittografico SHA3 [76]. Per cambiare il saldo dell'account,

dev'essere fornita la firma crittografica della chiave privata corrispondente alla chiave pubblica, in modo simile allo schema Pay-to-Public-Key-Hash di Bitcoin. Questa tipologia di account non supportano schemi di autenticazione avanzati.

Tabella B.1: Struttura di un account con proprietà esterna

EOA
Indirizzo
Saldo
Nonce

- **Contract account**: gli account di questo tipo B.1 sono identificati da un indirizzo. Tuttavia, l'indirizzo non è derivato da una chiave pubblica conosciuta; è invece derivato deterministicamente dall'indirizzo di un externally owned account conosciuto. La funzione crittografica SHA3 viene usato durante questo processo [77]. Come conseguenza, il saldo di un contract account non può essere controllato da una corrispondente chiave privata; invece, il contract account contiene del codice che può essere eseguito [78] e una struttura di dati persistente, una struttura che quando viene modificata conserva sempre la versione precedente di se stessa.

Transazioni

Similmente a Bitcoin, le transazioni Ethereum [79] possono inviare del valore da un *externally owned account* ad un altro. Come mostrato nella Tabella B.2, una transazione rappresenta il cambiamento di stato di un account identificato da un indirizzo. Una transazione è valida se il l'account mittente ha un saldo maggiore del *valore+commissioni* e se la transazione contiene la *transaction signature* (la firma della transazione) generata con la chiave privata del *externally owned account* mittente. Nei prossimi paragrafi verranno descritte con maggior dettaglio le commissioni sulle transazioni.

Tabella B.2: Struttura di una transazione

Transazione
recipient address
valore
dati
gas iniziale
prezzo del gas
transaction signature

Differentemente da Bitcoin, una transazione Ethereum eseguita da un externally owned account può creare nuovi contract account; questi vengono archiviati nella blockchain Ethereum. Infine, una transazione può essere inviata da un externally owned account ad un contract account esterno già esistente. La struttura della transazione contiene una transazione virtuale chiamata *message* (messaggio), la quale è memorizzata nei dati contenuti nella transazione, come mostrato dalla tabella 2.

B.3.2 Messaggi (messages)

I messaggi [80] sono delle informazioni che vengono scambiate fra gli accounts di entrambe le tipologie. Anche se i messaggi sono strutturalmente simili alle transazioni (Tabella B.2), questi non vengono archiviati nella blockchain come lo sono le transazioni. Un messaggio spedito da un externally owned account ad un contract account rappresenta il trasferimento di valore. Inoltre, in questo caso il messaggio invoca il codice contenuto nel contract account di destinazione. I messaggi sono analogicamente simili a delle funzioni o le procedure in programmazione.

Tabella B.3: Struttura di un messaggio [81]

Messaggio
indirizzo mittente
indirizzo ricevente
valore
dati
gas iniziale

In modo simile, i messaggi possono essere inviati da un contract account ad un altro contract account. In questo caso, l'analogia con le funzioni informatiche della programmazione è ancora più chiara.

B.3.3 Smart contracts

I contratti in Ethereum, detti *smart contracts* sono una raccolta di codice informatico e dati inclusi in un contract account nella blockchain di Ethereum [78]. Sebbene i contratti possano essere usati per implementare schemi di autorizzazione per spendere il saldo dell'account, come in Bitcoin, ci sono diverse significative differenze fra gli script di Bitcoin e i contratti di Ethereum.[18] Le caratteristiche generali dei contratti sono:

- quasi-Turing completi [80]
- non sono *stateless*, sono cioè dati informatici persistenti
- possono accedere ad alcuni dati degli altri contratti

Ethereum Virtual Machine (EVM)

L'Ethereum Virtual Machine è un ambiente di esecuzione (*runtime environment*) per i contratti di Ethereum. Il software EVM è incluso nei nodi della rete, i quali eseguono un contratto scritto in bytecode, un linguaggio intermedio fra il linguaggio macchina e il linguaggio di programmazione, usato per descrivere le operazioni che costituiscono

un programma. L'esecuzione del codice è isolata, il che significa che il codice che gira al suo interno non ha accesso al network, ai filesystem o ad altri processi; gli smart contracts hanno persino un accesso più limitato ad altri smart contracts. I contratti presenti sulla blockchain vengono registrati in uno specifico formato binario (EVM bytecode); tuttavia, i contratti sono generalmente scritti in un linguaggio *high level* di Ethereum, compilato in bytecode utilizzando l'EVM compiler e infine archiviati nella blockchain utilizzando un nodo della rete, detto anche Ethereum client. Ogni blocco di Ethereum contiene una *state root*, radice, di un *Patricia tree*. [82] La *state root* è un hash che contiene lo stato di tutti gli account e di ogni contratto esistente archiviato e incluso nel blocco. [83] Quando un miner individua un blocco candidato con incluse una serie di transazioni, la nuova *state root* viene calcolata processando tutte le transazioni nel blocco candidato. Questo processo esegue anche il codice presente negli smart contracts inclusi e può anche invocare il codice di altri contract accounts. Inoltre, lo stato di alcuni account può essere modificato e la *state root* viene ricalcolata. Questa viene poi inclusa nel blocco candidato e propagata agli altri nodi del network. Quando un nodo completo, *full-node*, riceve un nuovo blocco, la *state root* viene ricostruita localmente eseguendo tutte le transazioni nel blocco ricevente. Se la *state root* calcolata localmente è uguale alla *state root* inclusa nel blocco ricevuto, allora il blocco ricevuto è valido [84] e il processo di mining viene eseguito per ogni blocco. Anche se l'esecuzione di tutti i contratti viene eseguita da ogni *full-node* in modo indipendente e localmente, la ricompensa del blocco e le commissioni vengono accreditate solo al miner che propaga con successo il proprio *minted* (in italiano, coniato) blocco candidato nella blockchain globale.

Gas (costi di commissione) In Bitcoin, i miners prioritarizzano generalmente le transazioni con una commissione più alta, in modo tale da massimizzare il profitto. Ethereum include le commissioni per i miners per la stessa ragione di Bitcoin; tuttavia, esiste un'ulteriore ragione per pagare le *fees* dei miners. I contratti eseguiti nella EVM corrono il rischio di potenziali loop infiniti [85] o di eseguire operazioni che richiedono

un eccessivo sforzo computazionale. Per prevenire i DDoS attacks e ricompensare i miners per lo sforzo computazionale, i contratti consumano *gas*. Il gas è un unità interna utilizzata nella EVM; ogni istruzione impartita dalla EVM (*operation code*) ha assegnato un costo in gas (*gas cost*). Il costo del gas specifica quante unità di gas vengono consumate per esguire un operation code. Dopo l'esecuzione del programa, la EVM moltiplica il gas utilizzato in ogni singola operazione per il costo del gas. Se il gas utilizzato eccede il gas iniziale contenuto nella transazione o nel messaggio, l'esecuzione viene bloccata.

Tabella B.4: Istruzioni principali della EVM e il loro relativo costo

Nome dell'istruzione	Costo in gas	Descrizione
`STOP`	0	blocca l'esecuzione
`SUICIDE`	0	come STOP con anche la soppressione dell'account
`SHA3`	20	calcola un hash Keccak-256
`SLOAD`	20	estrae il primo elemento dello *stack*
`SSTORE`	100	salva le informazioni nell'archivio permanente
`BALANCE`	20	fornisce il saldo di un determinato account
`CREATE`	100	crea un nuovo account associato al codice
`CALL`	20	esegue una chiamata ad un account
`step`	1	rappresenta l'esecuzione di un ciclo
`memory`	1	ogni informazione addizionale che espande la memoria
`txdata`	5	dati in byte o codice per una transazione
`contract`	53000	crea un contratto da una transazione

I creatori di una nuova transazione devono stimare e includere in una transazione abbastanza gas iniziale necessario ad eseguire il codice del contratto nell'account di destinazione. Infatti, devono includere un *gas price* nella nuova transazione [81]. Il prezzo del gas non è altro che il prezzo offerto (*bid price*) in ETH per un'unità di gas; è possibile monitorare l'andamento del prezzo tramite ethgasstation.info, piattaforma che permette di visualizzare il costo medio attuale per eseguire una transazione sulla blockchain Ethereum. Solitamente i miners prioritarizzano le transazione con un *gas price* più alto; se l'esecuzione termina propriamente, il costo totale del gas utilizzato viene sottratto dal saldo dell'account che ha inviato la transazione. Quando non è

presente abbastanza gas per le istruzioni successive, il costo totale viene comunque sottratto dal bilancio dell'account sorgente e altre modifiche vengono effettuate mentre l'esecuzione viene invertita (e quindi arrestata).

Appendice C

Mercati decentralizzati

C.1 Background

Negli ultimi anni abbiamo assistito ad un sostanziale miglioramento e ad una rapida crescita nell'infrastruttura delle criptovalute, con gli exchanges (in italiano, borse finanziarie) che hanno avuto un ruolo da protagonista. Nel mercato delle valute digitali, un exchange di criptovalute non è altro che una piattaforma che permette la conversione di valuta fiat (cioè le monete a corso legale [86]) in criptovalute e viceversa, come anche la conversione di differenti coppie di criptovalute. Tuttavia, questi exchanges sono costantemente oggetto di hack[1]: ad inizio 2018, ad esempio, il secondo più grande exchange di criptovalute del Giappone, Coincheck, è stato hackerato, risultando in una perdita di circa 500 milioni di dollari per i suoi investitori [87]. I problemi di sicurezza legati a questi exchanges sono quindi risultati uno degli argomenti più discussi dalla critica e una delle maggiori preoccupazioni per gli investitori di criptovalute. Verranno quindi analizzati, in prima battuta, gli exchanges centralizzati di criptovalute, per poi

[1] Un hack è l'azione di qualcuno che cerca di violare le difese e sfruttare le debolezze di un computer o di una rete. Gli hacker possono essere motivati da una moltitudine di motivi, come il profitto, la protesta, la raccolta di informazioni, la sfida, il divertimento, o per valutare le debolezze di un sistema con lo scopo di contribuire a formulare difese contro potenziali hacker.

spostare l'attenzione ad una diversa categoria di exchange, la quale ha visto la propria popolarità aumentare nell'ultimo periodo: gli exchanges decentralizzati.

C.1.1 Exchanges centralizzati

Gli exchange centralizzati (da qui CEx) sono sicuramente la tipologia più diffusa nel mondo della finanza. Questi mercati permettono lo scambio di strumenti finanziari, come titoli azionari o criptovalute, fungendo da aggregatori di liquidità, custodendo gli assets dei clienti ed incrociando domanda ed offerta. Un esempio di exchanges decentralizzati sono il New York Stock Exchange (NYSE) e il London Stock Exchange (LSE). Ogni operazione eseguita si riflette in un cambiamento nel database dell'exchange, e gli utenti depositano i propri fondi in un conto (per quanto rigurda le criptovalute, nel wallet dell'exchange) che contiene i fondi di tutti i clienti dell'exchange. Questi infatti rientrano effettivamente in possesso dei loro fondi solamente nel momento in cui richiedono esplicitamente il ritiro (liquidazione???) dall'exchange.I CEx, per le criptovalute, risultano più veloci grazie, appunto, alla loro centralizzazione e al fatto che lo scambio degli assets non avviene on-chain, permettendo quindi di non dover attendere la validazione delle transazioni da parte dei miners. I benefici dei CEx includono anche una maggiore liquidità, la possibilità di utilizzare valuta fiat per gli investimenti, funzioni di trading avanzate e la possibilità di implementare sofisticate strategie di trading, offrono supporto agli investitori istituzionali e sono conformi con le regolamentazioni dello stato in cui hanno sede. Tuttavia, hanno una serie di punti di debolezza che rendono insicure queste piattaforme. C'è la possibilità che i fondi versati nell'exchange centralizzato possano essere custoditi in un hot" wallet nel server della piattaforma per facilitare il trasferimento dei fondi. Gli hot wallet sono una grande fonte di rischio per il fatto che sono costantemente online; se un CEx ha una struttura e delle misure di sicurezza deboli, diventa un bersaglio potenziale per gli hackers.

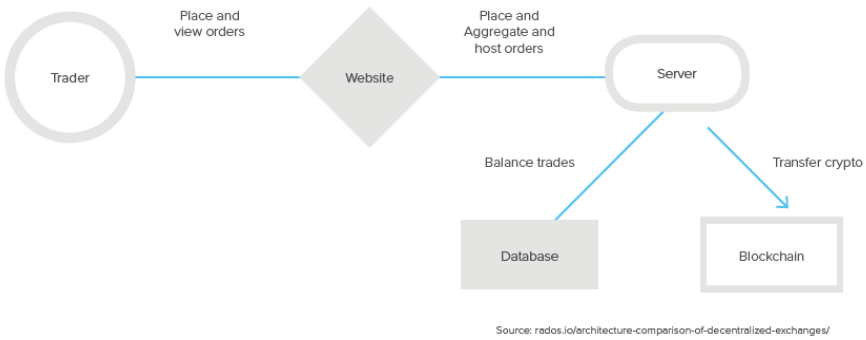

Figura C.1: Struttura degli exchange centralizzati

TOP FIVE EXCHANGE HACKS

Exchange	Date	Amount	Token stolen
MtGox	Mar-2014	$700,000,000	BTC
Coincheck	Jan-2018	$534,800,000	NEM
BitGrail	Feb-2018	$195,000,000	NANO
Bitfinex	Aug-2016	$72,000,000	BTC
Nicehash	Dec-2017	$60,000,000	BTC

Source: rados.io/list-of-documented-exchange-hacks/

Figura C.2: Perdite derivanti da hacks negli exchange centralizzati

Negli ultimi due anni i CEx sono diventati più prudenti nel quotare gli assests [88] per le lacune legislative e le leggi sulla sicurezza. Per questa ragione, si è intensificato lo sviluppo dei mercati decentralizzati (*Decentralized Exchanges*, chiamati anche *DExs*), con l'intento di risolvere alcune debolezze degli exchange centralizzati.

C.1.2 Exchanges decentralizzati

Gli exchange decentralizzati (DEXs) sono delle dApp sviluppate su piattaforme, fra cui Ethereum, che permettono l'esecuzione di smart contracts; questi vengono usati per gestire le transazioni. Ci sono circa 250 DExs [89] e oltre 30 protocolli DExs [90]. I protocolli DExs non sono esattamente una piattaforma d'investimento, ma piuttosto permettono ai teams di sviluppo uno strumento affidabile per programmare gli smart contracts necessari alla creazione e funzionamento degli exchanges. Un protocollo popolare (nel 2019) è 0x, di cui parleremo nella Sezione **??**. La caratteristica principale che rende i DEXs decentralizzati è che non hanno la proprietà degli assets dei clienti (*non-custodial*).

C.1.3 Off-chain orderbook e On-chain orderbook

Le differenze chiave riguardano principalmente dove viene effettuato l'*hosting*[2] dell'orderbook (*off-chain*, su un server centrale, o *on-chain*, come si vedrà nel prossimo paragrafo), come gli ordini vengono creati, modificati, accoppiati e cancellati, ed infine come le transazioni vengono eseguite. Queste determinano il grado di decentralizzazione dei DEXs. Per esempio, un DEX potrebbe essere completamente decentralizzato e archiviare ogni ordine creato, modificato, cancellato ed eseguito sulla blockchain.

Prendiamo ora in considerazione due DEXs che rientrano fra i primi cinque DEXs per volume, come riportato dalla Figura C.3. Questi DEXs sono *0x* e *IDEX*, ed hanno

[2]In informatica si definisce hosting (dall'inglese to host, ospitare) un servizio di rete che consiste nell'allocare su un server web delle pagine web di un sito web o di un'applicazione web, rendendolo così accessibile dalla rete Internet e ai suoi utenti.

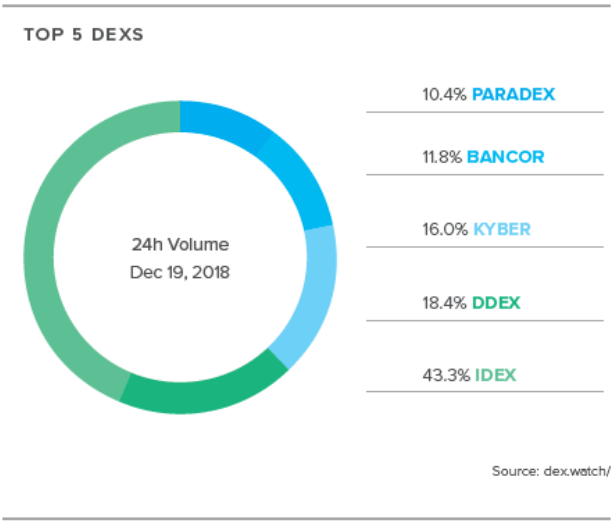

Figura C.3: Maggiori 5 DEXs per volumi giornalieri

in comune la caratteristica di essere degli exchange che utilizzano una struttura ibrida, nella quale l'orderbook viene mantenuto off-chain e l'effettiva esecuzione degli ordini avviene on-chain. Questo permette agli utenti di inserire e modificare gli ordini in real-time, in modo molto più veloce rispetto che se le transazioni fossero eseguite e registrate on-chain.
Con una struttura ibrida, si cerca di minimizzare la necessità di affidarsi ad un'autorità centrale; invece, gli investitori hanno la possibilità di scegliere ed eseguire gli ordini direttamente sulla blockchain, firmando le transazioni con la propria chiave privata.

C.1.4 Vantaggi principali dei DEXs

Tutti i DEXs sono *non-custodial*, e ciò significa che tali piattaforme non custodiscono i fondi dei loro utenti in un singolo wallet o in un server centrale. Invece, gli utenti sono in possesso dei loro fondi e autorizzano lo smart contract del DEX ad accedere ai loro fondi per effettuare le operazioni di trading. Questo permette di eliminare il rischio,

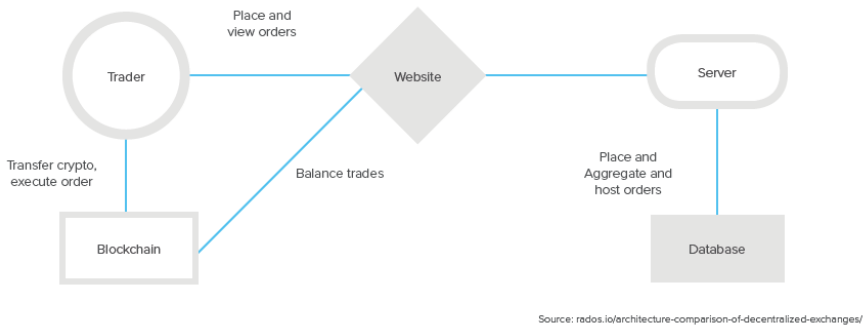

Figura C.4: Struttura degli exchange decentralizzati

legato alla sicurezza della piattaforma, di custodire gli assets dei clienti in un singolo wallet; i fondi, infatti, vengono scambiati tramite il modello del peer-to-peer.

I DEXs possono avere infinite integrazioni con altre dApps presenti sulla piattaforma Ethereum, le quali possono fornire ulteriori funzionalità per facilitare le operazioni di trading. Per esempio, il team che lavora sullo sviluppo di 0x ha dichiarato che i relayers avrebbero la possibilità di implementare il KYC[3] integrando fornitori di tali servizi come Polymath (`polymath.network`) o Harbor (`harbor.com`) per integrare una whitelist[4] degli indirizzi Ethereum, cioè una lista di indirizzi che hanno effettuato il processo KYC [91] e hanno ricevuto l'approvazione ad investire sulla piattaforma. Un'altra caratteristica peculiare è l'astrazione dei tokens, che permette di mettere in disparte la conversione fra token differenti per gli utenti della piattaforma[5].

[3]Con l'espressione *Know Your Customer* (in inglese "conosci il tuo cliente", spesso abbreviata in KYC) si intende un processo di riconoscimento utilizzato dalle aziende per verificare l'identità dei propri clienti e valutare potenziali rischi o intenzioni illegali nel rapporto con il cliente.

[4]Un elenco di persone o cose che una particolare entità o gruppo considera accettabili e di cui ci si deve fidare

[5]Un esempio di questa astrazione è Weasel (`devpost.com/software/weasel`), una dApp che permette ai possessori di ETH di inviare DAI tokens utilizzando Status, un servizio chat decentralizzato, senza che gli utenti debbano confertire gli ETH in DAI; Weasel scambia gli ETH per i DAI a nome dei propri utenti utilizzando la rete KyberNetwork.

C.1.5 Sfide principali dei DEXs

User experience

La user experience[6] (UX), cioè l'esperienza che ha un utente quando utilizza un prodotto o un servizio, dei DEXs (e della maggior parte delle dApps in generale) è tuttora in fase di sviluppo e miglioramento, in quanto le problematiche nella facilità d'uso sono ancora molte. Fra queste si può citare il fatto che gli utenti sono obbligati a gestire le operazioni di investimento attraverso il proprio wallet, piuttosto che utilizzare un account, la latenza associata alle transazioni on-chain rende ancora il tutto macchinoso e altro ancora. Gli exchanges centralizzati, infatti, hanno un interfaccia familiare e relativemente facile da utilizzare, mentre molti DEXs hanno un'interfaccia confusa e la user experience non è delle migliori.

Bugs negli smart contracts

Le DApp di Ethereum e i loro relativi smart contracts sono sviluppati utilizzando il linguaggio Solidity; tuttavia, è generalmente riconosciuto che Solidity presenta una serie di difetti strutturali e una forte mancanza di strumenti per testare e verificare gli smart contracts. Ciò comporta che sia presente il rischio di errori nello sviluppo degli smart contract, mettendo potenzialmente repentaglio i fondi degli utenti.

Funzionalità cross-blockchain I DEXs basati su Ethereum offrono la possibilità di utilizzare solamente i token ERC-20 e/o i token ERC-721 [**https://eips.ethereum.org/erc**], e non offrono alcuna possibilità di effettuare transazioni fra diverse blockchains. I cosiddetti *atomic swap* cross-blockchain [**https://arxiv.org/pdf/1801.09515.pdf**], cioè gli scambi istantanei di cryptovalute che non sono sulla stessa blockchain, sono una tecnologia sviluppata recentemente e che potrebbe risolvere tale mancanza nei DEXs. Tuttavia, i DEXs attuale si sono dimostrati riluttanti nell'offrire tale servizio. Infine, i

[6]Per esperienza d'uso (più nota come User Experience o UX) s'intende ciò che una persona prova quando utilizza un prodotto, un sistema o un servizio. L'esperienza d'uso concerne gli aspetti esperienziali, affettivi, l'attribuzione di senso e di valore collegati al possesso di un prodotto e all'interazione con esso, ma include anche le percezioni personali su aspetti quali l'utilità, la semplicità d'utilizzo e l'efficienza del sistema.

DEXs non offrono una metodo semplice e diretto per la conversione delle valute fiat in criptovalute; così, gli utenti sono obbligati a convertire il proprio denaro in ETH per poi convertirli nei token presenti sul DEX.

Attacchi DDos e DNS

Stando alla teoria, uno dei maggiori vantaggi delle dApps che vengono eseguite su blockchain è il fatto che hanno un downtime[7] pari a zero. Tuttavia, la presenza di una componente centralizzata comporta la perdita di tale vantaggio; più la struttura di un DEX risulta centralizzata, più alto sarà anche il rischio che essa sia target di un attacco DDos (Distributed Denial of Service) o di un attacco DNS (Domain Name Service) sul suo server centrale (questo però è un rischio presente anche negli exchange centralizzati). Un attacco DDoS avviene nel momento in cui il server viene sovracaricato di traffico proveniente da diverse fonti: ciò rende i servizi online non più disponibili. Un attacco DNS invece accade quando un aggressore prende il controllo del server DNS di un sito internet e reindirizza i visitatori ad un altro sito, controllato anch'esso dall'hacker, con l'intenzione di compromettere le informazioni e i fondi degli utenti[8].

Regolamentazione

Quando i primi DEX iniziarono ad operare, le persone assunsero che i benefici includevano anche la possibilità di non effettuare il KYC e che tali piattaforme eliminassero la possibilità di censura. Tuttavia, se una componente dei DEX può essere centralizzata, allora può essere anche sottoposta a regolamentazione. I legislatori hanno recentemente dimostrato che anche i DEXs che sono completamente decentralizzati potrebbero essere soggetti a regolamentazione se l'autorità di vigilanza ritiene che essi facilitino lo scambio di strumenti finanziari, fra cui anche opzioni o strumenti derivati, sottoforma

[7] Il termine downtime (traducibile in «tempo di fermo»), denota lo stato di un sistema che non è operativo oppure l'intervallo di tempo in cui un sistema è in tale stato, che può essere dovuto ad un guasto, a manutenzione o altre cause.

[8] EtherDelta è stato soggetto di un attacco DNS nel Dicembre del 2017 [92]: gli hacker non riuscirono ad entrare in controllo degli smart contracts che permettono all'exchange di funzionare, ma furono comunque in grado di rendere inutilizzabile il server DNS per diverse ore; inoltre, molti utenti del sito, durante l'attacco, mandarono inconsapevolmente i propri fondi all'indirizzo degli hackers. Si stima che circa $250'000 in ETH e una somma ancora sconosciuta di altri token ERC-20 siano stati inviati a questi aggressori.

di tokens. Nel novembre 2018, la SEC ha accusato il CEO di EtherDelta di aver violato la regolamentazione federale sugli strumenti finanziari (*Exchange Act Section 5*) per aver avviato un exchange non registrato. In accordo con la SEC, EtherDelta è stato qualificato come mercato regolamentato in quanto operava come marketplace che permette di aggregare gli ordini di acquisto e di vendita in tokens considerati alla pari di strumenti finanziari[93]. Nonostante ciò, la mancanza di una regolamentazione precisa in questo ambito, ha portato la SEC a considerare ad intermittenza i token come strumenti finanziari ufficiali, lasciando quindi ancora offuscata la loro legalità. Alcuni DEXs infatti stanno evitando di quotare gli assets che i regolatori considerano strumenti finanziari, in modo tale da non dover registrare la piattaforma come un mercato finanziario; altrimenti, nel caso in cui un DEX offra strumenti finanziari riconosciuti dall'autorità di vigilanza, l'unico modo per continuare ad operare sarebbe quello di essere registrati come Alternative Trading System (ATS) [94]. Se un DEX ha una struttura completamente decentralizzata (orderbook completamente on-chain e non-custodial) e offre dei tokens considerati strumenti finanziari, risulterà molto difficile per l'autorità di vigilanza imporre una regolamentazione in quanto non c'è un'autorità centrale come punto di riferimento. Questo comporta, per la parte legata alla regolamentazione dei DEXs, un cambiamento nelle metodologie legislative per poter comprendere anche una logica completamente decentralizzata. Altri DEXs invece hanno utilizzato un approccio più proattivo, implementando il KYC e il AML (Anti Money Laundering [95]) per poter risultare conformi con la regolamentazione attuale dei mercati finanziari. Un esempio di questi exchanges è IDEX [96], di cui parleremo nei prossimi paragrafi. 0x, inoltre, ha dichiarato che i relayers hanno la possibilità di limitare lo scambio di determinati tokens (chiamati "permissioned tokens" [97]) ai soli utenti che hanno completato un processo di KYC/AML off-chain attraverso entità come le già citate Harbor e Polymath.

Esecuzione e cancellazione on-chain

La latenza e i costi associati alla blockchain Ethereum creano un sfida per i DExs che

decidono di utilizzare una struttura con esecuzione e cancellazione degli ordini completamente on-chain. Le limitazioni attuali comportano l'impossibilità di implementare delle funzionalità aggiuntive per gli exchange e lasciano aperte ulteriori opportunità per gli investitori. Fra queste possiamo trovare:

- Strategie di trading: molti DExs non supportano ancora le sofisticate funzioni delle piattaforme di trading "tradizionali" come il margin trading, gli ordini stop-loss, l'high frequency trading o altre strategie di investimento in quanto limitati dalla lenta esecuzione e cancellazione degli ordini on-chain. IDEX risolve questo problema tenendo l'esecuzione e la cancellazione separata dalla registrazione delle operazioni on-chain.

- Arbitraggio: i DEXs con cancellazione degli ordini on-chain sono più esposti ad avere opportunità di arbitraggio in quanto gli utenti non possono cancellare velocemente gli ordini in risposta alle fluttuazioni del mercato.

- Race conditions: l'esecuzione e/o la cancellazione on-chain degli ordini espone gli utenti anche al rischio di frontrunning, trade collision o maker griefing [98]

Liquidità

I DExs hanno, attualmente (nel 2019), una liquidità inadeguata e sono impelagati in un classico circolo vizioso che non porta benefici a nessuna delle parti: i traders sono attratti dalle piattaforme che offrono liquidità ma le piattaforme hanno bisogno di traders per aggregare la liquidità. Ad inizio febbraio 2019, i primi cinque DEXs hanno accumulato un volume di circa $2'175'000 contro i $3'558'900'000 in volume aggregato per i primi cinque exchange centralizzati. Tuttavia, i DEXs sono degli strumenti ancora in fase di sviluppo, e potrebbero incrementare la loro liquidità risolvendo le varie sfide elencate nei precedenti paragrafi.

Nelle prossime sezioni verranno analizzati in modo più dettagliato due dei tre DExs utilizzati da `AtOm`: Bancor e Uniswap.

C.2 Bancor Protocol: Automated Liquidity Provider

Bancor Protocol supporta una varietà di architetture per gli Smart Tokens, come i Liquidity Tokens e i Relay Tokens. I primi possono essere comprati o vendute in cambio dei tokens ai quali sono connessi ad un prezzo continuativamente calcolato; questo rimuove la necessità di trovare il match fra gli utenti, oltre che a minimizzare (o eliminare) il rischio di controparte e disassociare la liquidità dai volumi degli investimenti e dalle quotazioni negli exchange.

La liquidità detenuta dagli utenti rappresenta i vari saldi dei tokens, i quali sono depositati e posseduti dai creatori degli *smart tokens*, in modo tale da essere usati per eseguire delle transazioni basate sui dei prezzi aggiustati dinamicamente dalla formula. I saldi dei tokens forniscono una pool di liquidità condivisa che offre un'alternativa all'utilizzo di un orderbook per accoppiare gli ordini maker e taker[9].

I Relay Tokens hanno la funzione di gateway, cioè un collegamento, per la liquidità del network di Bancor. Questo permette ad ogni token di essere integrato all'interno della rete senza dover essere modificato per soddisfare i requisiti del Bancor Protocol; in questo modo, gli utenti possono scambiare il nuovo token con ogni token presente nel network di Bancor.

[9] Un "maker" è un utente che inserisce un ordine che non viene abbinato immediatamente, quindi rimane nel book e attende che qualcuno più tardi lo esegua, accettando la proposta. Il "taker" è un utente che decide di inserire un ordine che viene abbinato immediatamente con uno già esistente all'interno dell'order book. Le tariffe riservate ai maker sono generalmente più basse rispetto a quelle dei taker. Il motivo dietro a questo schema è la funzione del "maker" che fornisce liquidità nell'order book (inserendo una proposta di acquisto o di vendita che potrebbe essere abbinata in futuro, sta attivamente "creando" il mercato; mentre un "taker" consuma la liquidità del book "prendendo" un ordine, accettando quindi la proposta di un maker).

C.2.1 Vantaggi degli Smart Tokens

Gli Smart Tokens possono essere considerati un nuovo paradigma nel mercato dei tokens in quanto incorporano sia una componente autonomatizzata che una componente decentralizzata che permettono effettuare conversioni valutarie riflettendo matematicamente la domanda e l'offerta, adattando la dimensione della conversione in tempo reale. Questo introduce una serie di vantaggi rispetto al trading negli exchange tradizionali:

- Liquidità continua: gli utenti possono sempre comprare o vendere i tokens nel network utilizzando direttamente lo smart contract, anche quando non ci sono altri compratori o venditori nel mercato. Siccome il prezzo si aggiusta in base alla dimensione della conversione richiesta, ci sarà sempre un prezzo al quale il token potrà essere convertito. Questo vantaggio comporta, come già accennato, la disconnessione della liquidità dai volumi degli scambi.

- Zero commissioni: per default, gli Smart Tokens non applicano nessuna commissione per la conversione che eseguono; le uniche fees pagate dagli utenti sono quelle richieste dalla transazione per essere validata dalla blockchain (per esempio, il gas in Ethereum). I creatori di Smart Token hanno la possibilità di stabilire una commissione (chiamata *contribution*) per l'utilizzo del loro particolare Smart Token, ma questa sarà quasi certamente molto bassa per la natura open-source del protocollo: un altro utente infatti potrà facilmente creare un ulteriore Smart Token, con la medesima funzione del concorrente ma con una fee più bassa. Il Bancor Protocol non richiede alcun tipo di commissioni, ma trae piuttosto beneficio dall'incremento dell'adozione e dall'aumento degli utenti e del numero di tokens nel network.

- Sensibilità del prezzo adattabile: grazie al rapporto fisso fra le due liquidità impostato nello Smart Token rende il suo prezzo meno semsibile alle speculazioni di breve periodo o alle turbolenze causate dagli ordini di grandi dimensioni. Per

esempio, uno Smart Token con un rapporto del 10% è comparabile ad un exchange con un orderbook uguale, in termini di valore, al 10% della capitalizzazione di mercato totale del token. Questa sensibilità può essere aggiustata modificando il rapporto, con lo scopo di raggiungere il comportamento desiderato per un particolare Smart Token.

- Nessuno spread bid-ask: La formula di Bancor applica lo stesso metodo di calcolo sia per gli ordini di acquisto sia per gli ordini di vendita. Questa è una differenza rilevante rispetto agli exchanges tradizionali dove il bid è sempre minore dell'ask. Lo spread bid-ask, come abbiamo visto nei capitoli precedenti, è ciò che permette ai market makers di ottenere un profitto. Il Bancor Protocol invece non richiede un profitto per mantenersi operativo; potrebbe essere introdotto uno spread decentralizzato per incoraggiare l'adozione del network, beneficiando tutti i partecipanti.

- Prezzo prevedibile: l'algoritmo del prezzo di uno Smart Token è completamente trasparente, permettendo agli utenti di calcolare anticipatamente il prezzo effettivo della conversione valutaria desiderata, prima della sua esecuzione. Questa è un'ulteriore differenza con gli orderbook tradizionali, dove un ordine di grandi dimensioni può causare un movimento imprevedibile del prezzo verso un livello significativamente diverso.

- Basati su ERC20: gli Smart Tokens sono dei token ERC20-compliant, e quindi possono essere integrati senza alcuna difficoltà in altre dApps o wallet Ethereum, in quanto sono conformi ai protocolli standard di Ethereum. Infine, ogni token ERC20 esistente può essere connesso al Bancor Network attraverso uno Smart Token, rendendo Bancor una piattaforma perfettamente allineata con l'ecosistema Ethereum.

C.2.2 Ecosistema di Bancor

Il successo del Bancor Network nel fornire una liquidità decentralizzata dipende anche dalla partecipazione di una varietà di utenti differenti. L'ecosistema di Bancor è quindi composto da:

- Investitori: sono gli utenti finali che custodiscono, convertono e trasferiscono Smart Tokens
- Creatori di Smart Tokens: sono persone, società, comunità, organizzazioni o fondazioni che emettono nuovi Smart Tokens, configurandone il volume iniziale, il prezzo, il rapporto fra i due token convertiti e gestendo l'emissione iniziale dei tokens. Questi includono anche i creatori di Relay Tokens, i quali possono connettere qualsiasi ERC20 token esistente al network.
- Asset Tokenizers: sono i creatori dei Relay Tokens, i quali permettono di rappresentare un asset reale o un token di un'altra blockchain. Questo permette agli Smart Tokens di connettersi ad un ampio paniere di assests, come il Bitcoin, la valuta fiat, l'oro o altre blokchain.
- Arbitraggisti: sono gli investitori che monitorano la liquidità del network per trovare dei prezzi inconsistenti sia con quelli di exchange esterni che quelli di altri Smart Tokens, per poi ripristinare la consistenza del prezzo tramite l'arbitraggio, come spiegato nei precedenti capitoli. Gli arbitraggisti sono naturalmente incentivati a mantere i prezzi consistenti e pertanto hanno un ruolo fondamentale dell'ecosistema di Bancor.

C.3 UniSwap: un esempio

Come già analizzato nella Sezione 4.4.2, Uniswap è un mercato decentralizzato che determina i prezzi automaticamente, utilizzando il cosiddetto meccanismo del *constant market maker* ($x \times y = k$) [43], che mantiene le riserve complessive di tokens in un equilibrio relativo; le riserve vengono formate inizialmente da una rete di fornitori di liquidità[10] che forniscono tokens al sistema in cambio di una quota proporzionale delle commissioni sulle transazioni. Verrà ora analizzato un esempio del funzionamento di Uniswap.

L'esempio mostra uno scambio fra ETH e OMG (un token ERC20); $10ETH$ e $500OMG$ vengono depositati in uno smart contract di Uniswap (che li inserisce nelle pool[11]). Una costante (k) viene impostata automaticamente in modo che $ETH_{pool} * OMG_{pool} = k$. Nell'esempio, le formule indicheranno la quantità di tokens contenuti in ogni pool.

$$ETH_{pool} = 10$$

$$OMG_{pool} = 500$$

$$k = 10 \times 500 = 5000$$

Un investitore intenzionato ad acquistare OMG invia $1ETH$ al contratto. Viene prelevata una commissione dello 0,25% per i fornitori di liquidità presenti su Uniswap;

[10] Uniswap permette agli utenti di fornire automaticamente liquidità ad un mercato, dividendo i fondi allocati al 50% per entrambi i token (ETH-ERC20 o ERC20-ERC20) che compongono l'asset; con un mercato senza oscillazioni, è possibile ottenere come profitto le commissioni sulle operazioni.

[11] Le pool di liquidità sono una sorta di market makers autonomi, chiamati *Constant Product Market Maker* [99], il cui funzionamento è permesso da uno smart contract. Essi forniscono costantemente il 50% dei fondi allocati al mercato ed ottengono, in un mercato stabile, un interesse rappresentato dalla commissione prelevata per i *trades*.

vengono quindi aggiunti $0,9975ETH$ alla ETH_{pool} e, successivamente, la costante viene divisa per la nuova quantità di ETH nella pool di liquidità per determinare la nuova OMG_{pool}. I token *OMG* rimanenti vengono inviati all'acquirente.
L'acquirente invia $1ETH$.

$$Fee = 1ETH/500 = 0,0025ETH$$

$$ETH_{pool} = 10 + 1 - 0,0025 = 10,99975$$

$$OMG_{pool} = 5000/10.9975 = 454,65$$

L'acquirente riceve in cambio $500 - 454,65 = 45,35OMG$.

La commissione viene quindi ricollocata nel pool di liquidità, e verrà suddivisa fra i vari fornitori di liquidità disponibili per il mercato. Poiché la commissione viene aggiunta dopo il calcolo del prezzo, la costante aumenta leggermente con ogni operazione, rendendo il sistema redditizio per i fornitori di liquidità. Infatti, ciò che k rappresenta realmente è $ETH_{pool} \times OMG_{pool}$ al termine del precedente scambio.

$$ETH_{pool} = 10,9975 + 0,0025 = 11$$

$$OMG_{pool} = 454,65$$

$$k_{new} = 11 * 454,65 = 5001,15$$

In questo caso l'acquirente ha acquistato a 44,5 OMG/ETH. Dopo lo scambio, il prezzo è variato. Se un altro acquirente effettua l'acquisto dello stesso asset (OMG nell'esempio), otterrà un prezzo leggermente più elevato per OMG/ETH. Tuttavia, se un acquirente effettua un acquisto di ETH, otterrà un prezzo ETH/OMG leggermente più basso.

$$Input : 1ETH$$

$$Output : 44.5OMG$$

$$Prezzo = 44,5OMG/ETH$$

Gli acquisti di grandi dimensioni, soprattutto se più alti rispetto alla dimensione totale della pool di liquidità, causeranno uno slippage del prezzo. In un mercato attivo, l'arbitraggio può permettere di mantenere il livello dello slippage basso, ed allineato fra i diversi DExs.

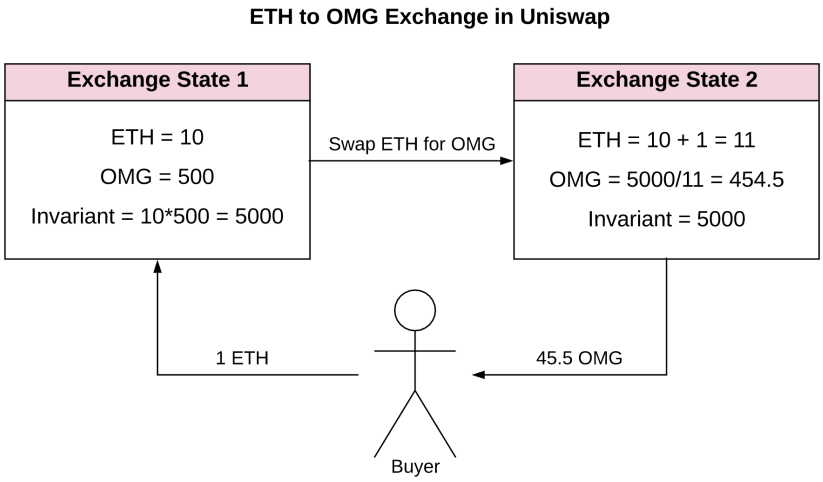

Figura C.5: Schema grafico del esempio descritto nella Sezione C.3

Bibliografia

[1] Nick Szabo. «Formalizing and securing relationships on public networks» (1997).

[2] 2018. URL: https://vision.visaeurope.com/blogs/sweden-in-2018-one-step-closer-to-cashless.

[3] 2017. URL: https://www.guidafisco.it/pagamenti-contanti-limite-massimo-1000-euro-615.

[4] 2019. URL: https://moneymattersforglobetrotters.com/international-transactions-101-credit-card/.

[5] Aurelio F. Bariviera et al. «Some stylized facts of the Bitcoin market». *Physica A: Statistical Mechanics and its Applications* 484 (2017), pp. 82–90. URL: http://www.sciencedirect.com/science/article/pii/S0378437117304697.

[6] 2019. URL: https://coinmarketcap.com/.

[7] Satoshi Nakamoto. «Bitcoin: A Peer-to-Peer Electronic Cash System». *www.bitcoin.org* (2008).

[8] Florian Glaser et al. «Bitcoin - Asset or currency? Revealing users' hidden intentions». *Goethe University* (2013).

[9] Anne Haubo Dyhrberg. «Hedging Capabilities of Bitcoin. Is it the virtual gold?» *EconStor* (2015).

[10] European Central Bank. «Virtual Currency Schemes». *Eurosystem* (2012).

[11] 2019. URL: https://www2.deloitte.com/insights/us/en/topics/understanding-blockchain-potential/global-blockchain-survey.html?icid=dcom_promo_featured%7Cus;en.

[12] Juho Lindman, Virpi Tuunainen e Mattia Rossi. «Opportunities and Risks of Blockchain Technologies: A Research Agenda» (2017).

[13] Workie & Jain. «Distributed ledger technology: Implications of blockchain for the securities industry». *FINRA* (2017).

[14] Gareth W. Peters e Efstathios Panayi. «Understanding Modern Banking Ledgers through Blockchain Technologies: Future of Transaction Processing and Smart Contracts on the Internet of Money». *arXiv* (2015).

[15] Ralph C. Merkle. «A Certified Digital Signature». *BNR Inc.* (1979).

[16] Daniel Drescher. *Blockchain Basics: A Non-Technical Introduction in 25 Steps*. Apress, 2017.

[17] Arvind Narayanan et al. «Bitcoin and Cryptocurrency Technologies». *Princeton University Press* (2016).

[18] Buterin Vitalik. «Ethereum white paper» (2014). URL: https://github.com/ethereum/wiki/wiki/White-Paper.

[19] Thibaut Sardan. «What is a light client and why you should care?» (2018). URL: https://www.parity.io/what-is-a-light-client.

[20] Chris Dannen. *Introducing Ethereum and Solidity Foundations of Cryptocurrency and Blockchain Programming for Beginners*. Apress, 2017.

[21] Wood Gavin. «Ethereum Yellow Paper: a secure decentralized generalised transaction ledger». *Ethereum & Parity* (2019).

[22] URL: https://solidity.readthedocs.io/en/v0.5.1/.

[23] Philip Daian et al. «Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges». *Cornell Tech University* (2019).

[24] Henry Berg e Todd Proebsting. «Hanson's Automated Market Maker». *Journal of Prediction Markets* 3 (2009), pp. 45–59.

[25] URL: https://etherdelta.com/.

[26] https://0xtracker.com/relayers.

[27] Imad A. Moosa. *International Financial Operations: Arbitrage, Hedging, Speculation, Financing and Investment*. Palgrave McMillan, 2003.

[28] Edward O. Thorp. *The Kelly Criterion In Blackjack Sports Betting, and the Stock Market*. Elsevier B.V, 2006.

[29] 2011. URL: https://www.milanofinanza.it/news/i-coefficienti-alpha-e-beta-201007281039564639.

[30] Kuepper Justin. «Trading The Odds With Arbitrage» (2019). URL: https://www.investopedia.com/articles/trading/04/111004.asp.

[31] 2005. URL: http://people.stern.nyu.edu/adamodar/pdfiles/invphiloh/arbitrage.pdf.

[32] Valerio Vallefuoco. «Fisco e Bitcoin, ipotesi tassazione sui proventi da capital gain». *Il Sole 24 Ore* (2018).

[33] 2013. URL: https://help.finecobank.com/it/mercati-e-trading/tobin-tax.html.

[34] Marios John Mavrides. *The efficiency of triangular arbitrage in the foreign exchange market*. University of Illinois at Chicago, 1991.

[35] Miltiades Chacholiades. «The Sufficiency of Three-Point Arbitrage to Insure Consistent Cross Rates of Exchange». *Southern Economic Journal* (1971).

[36] Andrei Shleifer e Robert W. Vishny. «The Limits of Arbitrage». *The Journal of Finance* (1997).

[37] 2010. URL: https://www.ilsole24ore.com/art/SoleOnLine4/100-parole/Economia/M/mark-to-market.shtml?uuid=ec917804-580a-11dd-93cb-a54c5cfcd900&DocRulesView=Libero&refresh_ce=1.

[38] 2019. URL: https://www.ig.com/it/glossario-trading/definizione-di-ordine-a-mercato.

[39] 2019. URL: https://web3js.readthedocs.io/en/1.0/getting-started.html.

[40] Galia Benartzi Eyal Hertzog Guy Benartzi. «Bancor Protocol WhitePaper». *Bancor Protocol* (2018).

[41] 2018. URL: https://support.bancor.network/hc/en-us/articles/360000462671-How-to-convert-tokens-on-the-Bancor-Web-App.

[42] 2018. URL: https://blog.bancor.network/rethinking-the-order-book-the-march-towards-automated-markets-150f1325fb8c.

[43] Daejun Park Yi Zhang Xiaohong Chen. «Formal Specification of Constant Product ($x \times y = k$) Market Maker Model and Implementation». *Runtime Verification, Inc.* (2018).

[44] 2019. URL: https://docs.ethhub.io/.

[45] 2019. URL: https://www.statista.com/statistics/647523/worldwidebitcoinblockchainsize/2019.

[46] Andreas M. Antonopoulos. *Mastering Bitcoin: Programming the Open Blockchain*. O'Reilly Media; 2 edition, 2017.

[47] 2019. URL: https://homes.esat.kuleuven.be/~bosselae/ripemd160.html.

[48] 2019. URL: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf.

[49] 2019. URL: https://github.com/bitcoin/bitcoin/blob/57704499be948c640c789c7fc11ed1abf8a
src/base58.cpp#L117.

[50] https://en.bitcoin.it/wiki/Protocol_rules. 2019.

[51] 2016. URL: https://en.wikipedia.org/wiki/Halting_problem.

[52] 2013. URL: www.bitcoin.org/en/glossary/pubkey-script.

[53] 2013. URL: www.bitcoin.org/en/glossary/signature-script.

[54] 2013. URL: https://bitcoin.org/en/glossary/standard-transaction.

[55] John Proos e Christof Zalka. «Shor's discrete logarithm quantum algorithm for elliptic curves». *University of Waterloo* (2004).

[56] 2016. URL: www.bitcoin.org/en/glossary/consensus-rules.

[57] 2016. URL: https://bitcoin.org/en/developer-guide#term-highprioritytransactions.

[58] 2016. URL: https://www.bitcoinclock.com/.

[59] 2016. URL: https://en.bitcoin.it/wiki/Controlled_supply.

[60] 2016. URL: https://en.bitcoin.it/wiki/Proof_of_work.

[61] 2016. URL: https://en.bitcoin.it/wiki/Hashcash.

[62] 2008. URL: https://satoshi.nakamotoinstitute.org/emails/cryptography/5/.

[63] 2013. URL: https://en.bitcoin.it/wiki/Protocol_rules#.22block.22_messages.

[64] 2013. URL: https://bitcoin.org/en/glossary/confirmation-score.

[65] 2013. URL: https://en.bitcoin.it/wiki/Double-spending.

[66] 2013. URL: https://en.bitcoin.it/wikiWeaknesses#Attacker_has_a_lot_of_computing_power.

[67] Massimo Bianca. *Il contratto*. Giuffrè, Milano, 2000.

[68] 2015. URL: http://iang.org/papers/intersection_ricardian_smart.html.

[69] 2015. URL: http://webfunds.org/guide/ricardian.html.

[70] 2016. URL: http://financialcryptography.com/mt/archives/001595.html.

[71] 2015. URL: http://iang.org/papers/ricardian_contract.html.

[72] Alex Tapscott. *The Blockchain Revolution: How the Technology Behind Bitcoin is Changing Money, Business, and the World*. Vol. pp. 72, 83, 101, 127. Penguin, mag. 2016.

[73] 1997. URL: http://archive.is/20DNf.

[74] 2014. URL: https://en.bitcoin.it/wiki/contract#example_2:_escrow_and_dispute_mediation.

[75] https://blog.ethereum.org/2014/04/10/the-issuance-model-inethereum/. 2014.

[76] 2017. URL: http://ethereum.stackexchange.com/a/36192017.

[77] 2017. URL: http://ethereum.stackexchange.com/a/7612016.

[78] 2017. URL:http://ethdocs.org/en/latest/contractstransactions/contracts.html2016.

[79] 2017. URL: http://ethereum.stackexchange.com/a/2097.

[80] 2015. URL: http://paper.gavwood.com.

[81] 2019. URL:http://ethdocs.org/en/latest/contractstransactions/accounttypesgastransacti
html.

[82] 2015. URL: https://github.com/ethereum/wiki/wiki/patriciatree.

[83] 2015. URL: https://blog.ethereum.org/2015/06/26/statetreepruning/.

[84] 2015. URL: https://forum.ethereum.org/discussion/2324/howiscontractvalidated.

[85] 2019. URL: https://en.wikipedia.org/wiki/Halting_problem.

[86] Andrea Terzi. *La moneta*. Il Mulino, 2002.

[87] 2018. URL: http://fortune.com/2018/01/31/coincheckhackhow/.

[88] 2019. URL: https://news.bitcoin.com/cryptocurrency-exchanges-delist-dozens-of-struggling-altcoins/.

[89] 2019. URL: https://github.com/distribuyed/index.

[90] 2019. URL: https://github.com/evbots/dexprotocols.

[91] 2019. URL: https://www.knowyourcustomer.com/.

[92] 2019. URL: https://www.ccn.com/cryptocurrency-exchange-etherdelta-hacked-in-dns-hijacking-scheme/.

[93] 2018. URL: https://www.sec.gov/news/pressrelease/2018258.

[94] 2018. URL: https://www.sec.gov/foia/docs/atslist.htm.

[95] 2018. URL: https://www.sec.gov/about/offices/ocie/amlsourcetool.htm.

[96] 2018. URL: https://medium.com/aurora-dao/pragmatic-decentralization-how-idex-will-approach-industry-regulations-8b109212128a.

[97] 2018. URL: https://blog.0xproject.com/compliantpeertopeertrading4dab8e5c3162.

[98] 2018. URL: https://github.com/ConsenSys/0x-review/blob/master/report/3_general_findings.md.

[99] 2018. URL: https://docs.uniswap.io/frontend-integration/pool.