Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

UNA TECNICA PER L'ANTICONTRAFFAZIONE BASATA SU SISTEMA BLOCKCHAIN

Relatore:
Chiar.mo Prof.
Cosimo Laneve

Correlatore:
Dott.ssa
Adele Veschetti

Presentata da:
Daniele Morotti

Sessione I
Anno Accademico 2020 - 2021

Abstract

In questo elaborato verranno illustrate inizialmente le caratteristiche principali della **blockchain** e il fenomeno della contraffazione, con i vari metodi utilizzati al giorno d'oggi. In seguito verranno illustrati e confrontati i vari sistemi anticontraffazione dello stato dell'arte e quello proposto in questo elaborato. Infine verrà illustrata nel dettaglio l'implementazione del codice degli smart contract e verranno indicati alcuni test eseguiti sul sistema. La tecnologia blockchain potrebbe contrastare efficacemente il fenomeno della contraffazione in quanto permette di salvare informazioni nei suoi blocchi e, una volta avvenute le transazioni, i dati non possono essere alterati per via di alcune proprietà di sicurezza della struttura. Nel sistema implementato si salvano alcune informazioni relative ai prodotti, dal momento in cui sono stati creati in una fabbrica autorizzata fino al momento della consegna al consumatore, per tracciare eventuali manomissioni o irregolarità.

Indice

Capitolo 1

Introduzione

È stato implementato un sistema anticontraffazione utile per tracciare i capi d'abbigliamento e garantirne l'autenticità dalla fabbrica fino alla consegna in negozio. Si è deciso di concentrarsi sulla risoluzione di questo problema per via della crescita esponenziale di prodotti contraffatti presenti sul mercato.
Per contraffazione di un capo d'abbigliamento si intende la produzione di un manufatto molto simile all'originale. La sola lettura dell'etichetta dettagliata non potrà garantirci l'eventuale originalità dello stesso in quanto viene spesso riprodotto nei minimi dettagli [1]. La contraffazione sta crescendo vertiginosamente negli ultimi anni e causa perdite ai titolari dei diritti di proprietà intellettuale di centinaia di miliardi di dollari ogni anno. Un report dell'OCSE ha mostrato come nel 2016 siano stati importati beni contraffatti per un totale di 509 miliardi, circa il 3,3% delle importazioni globali di quell'anno. Il volume di scambi internazionali di merce contraffatta, in accordo con Global Brand Counterfeiting Report 2018, ha raggiunto 1,2 trilioni di dollari nel 2017 e ci si aspetta una crescita decisa con il passare degli anni.

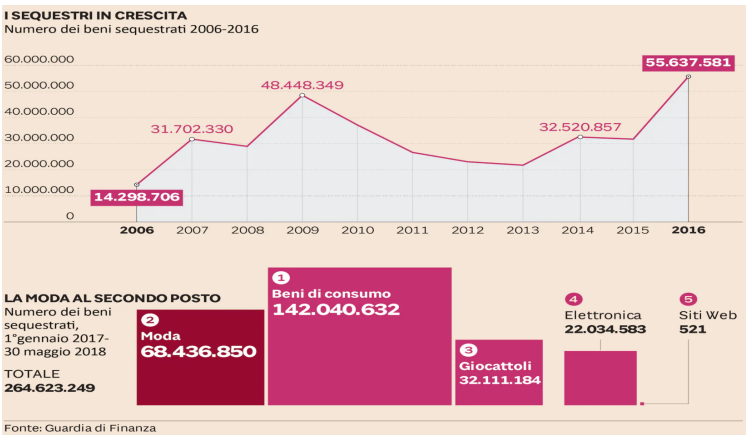

Figura 1.1: Dati sui beni sequestrati in Italia

La tesi è strutturata come segue, nel primo capitolo viene spiegata velocemente la blockchain con i suoi punti di forza e le sue debolezze e vengono illustrati i principali algoritmi di consenso che possono essere utilizzati. Nel secondo capitolo si illustrano varie tipologie di contraffazione, i metodi utilizzati dai contraffattori e lo stato dell'arte per quanto riguarda i sistemi anticontraffazione già sviluppati o in via di sviluppo.
Nel terzo capitolo si analizza l'efficacia del sistema proposto contro vari attacchi possibili e si esegue un confronto con gli altri sistemi dello stato dell'arte per evidenziare pregi e difetti del progetto. Nel quarto capitolo si descrivono l'ambiente di sviluppo utilizzato, tutti gli smart contract che fanno parte del sistema e i test effettuati per verificarne il funzionamento. Infine si discute del risultato raggiunto e di eventuali miglioramenti che si possono applicare al sistema.

1.1 La tecnologia blockchain

La blockchain è una struttura dati condivisa ed immutabile. È definita come un registro digitale aperto e distribuito in grado di memorizzare record di dati (solitamente denominati "transazioni") in modo sicuro, verificabile e permanente. Una volta scritti, i dati in un blocco non possono essere retroattivamente alterati senza che vengano modificati tutti i blocchi successivi ad esso e ciò, per la natura del protocollo e dello schema di validazione, necessiterebbe del consenso della maggioranza della rete. La blockchain fa parte della famiglia delle *Distributed Ledger*, ossia sistemi che si basano su un registro distribuito.

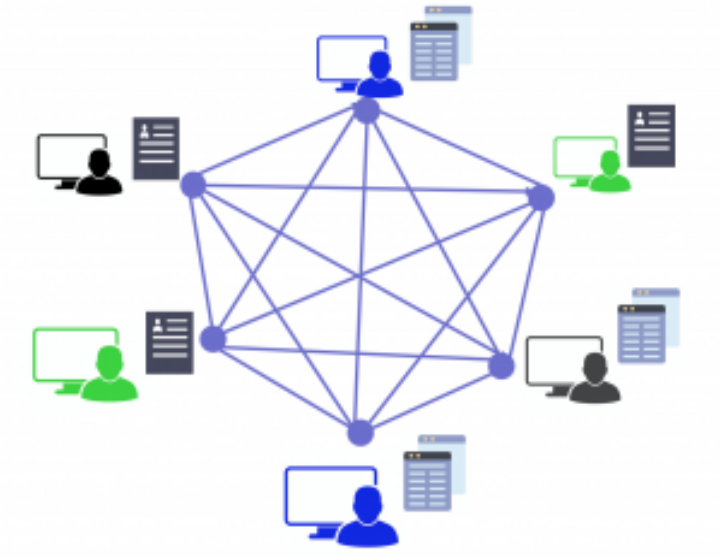

Figura 1.2: Registro distribuito (Fonte: Blockchain4innovation)

La grande sicurezza e affidabilità della blockchain sono legate all'utilizzo della crittografia sia per concatenare i blocchi tra loro che per firmare ogni transazione e confermare l'identità degli utenti coinvolti evitando, tramite l'utilizzo di codici univoci e timestamp, la possibile replica delle transazioni. Quando un nuovo blocco viene aggiunto alla blockchain, esso viene collegato al corrispettivo blocco precedente, utilizzando una funzione hash crittografica generata a partire dal blocco precedente. Nel caso in cui si volesse cambiare un blocco sarebbero da ricalcolare tutti gli hash dei blocchi successivi a quello e servirebbe il consenso della maggioranza della rete. Come anticipato, qualunque transazione è sottoposta ad un meccanismo di firma a doppia chiave asimmetrica che però non dipende da un organo certificatore centrale, proprio per mantenere una struttura decentralizzata.
I componenti principali della blockchain sono:

- Nodi: sono i partecipanti alla blockchain e sono costituiti fisicamente dai server di ciascun partecipante. Ognuno possiede una copia del registro.
- Blocchi: sono il raggruppamento di un insieme di transazioni che sono unite per essere verificate, approvate e poi archiviate dai partecipanti alla blockchain.
- Ledger: è il registro pubblico nel quale vengono "annotate" con la massima trasparenza e in modo immutabile tutte le transazioni effettuate in modo ordinato e sequenziale. Il Ledger è costituito dall'insieme dei blocchi che sono tra loro incatenati tramite una funzione di crittografia e grazie all'uso di hash.
- Transazioni: sono costituite dai dati che rappresentano i valori oggetto di "scambio", gli indirizzi pubblici delle persone coinvolte e la firma per garantire l'autenticità della transazione.

Esistono vari algoritmi di consenso utilizzati per garantire che tutti i nodi del sistema possano concordare sulla validità dei blocchi e sullo stato attuale della blockchain, anche nel caso in cui alcuni nodi fossero malevoli (*Byzantine fault tolerance*) o fallissero nella verifica.

1.1.1 Algoritmi di consenso

Per prima cosa si identificano gli aspetti che accomunano gli algoritmi di consenso.
Ad un nodo intenzionato a diventare un generatore di blocchi viene chiesto di fornire delle risorse (*stake*), in questo modo nel caso in cui dovesse imbrogliare perderebbe la quota versata. Per motivare i nodi alla partecipazione viene fornita una **ricompensa** a chi genererà il prossimo blocco, solitamente la ricompensa è composta dalle commissioni pagate da altri utenti, dalle unità di criptovaluta appena generate o da entrambi. Infine si deve essere in grado di scoprire quando qualcuno sta imbrogliando. Idealmente, dovrebbe

essere costoso produrre blocchi ma economico per chiunque **verificarli**. A seguire verranno elencati gli algoritmi più conosciuti ed utilizzati.

Proof of Work (PoW)

È stato implementato per la prima volta in Bitcoin ma il concetto è stato concepito molti anni prima. Questo algoritmo prevede che il *miner*, colui che crea il blocco, debba risolvere un problema matematico complesso, utilizzando quindi la propria potenza computazionale. Tutti i possibili miner devono quindi trovare un numero giusto tale per cui l'hash del blocco successivo inizi con un numero di zeri che varia in base alla difficoltà della rete in quel momento. La difficoltà del problema dipende dal numero di utenti, dalla potenza di calcolo disponibile e dal carico della rete. Una volta che un miner avrà trovato la soluzione al problema, la diffonderà agli altri nodi che verificheranno la correttezza e nel caso in cui essi raggiungano il consenso, il miner convaliderà il blocco aggiungendolo alla blockchain e guagagnerà una ricompensa.
I principali svantaggi sono i **costi elevati**, in quanto il processo di mining consuma molta energia elettrica e servono macchine specializzate e l'**inutilità** dei calcoli eseguiti, che servono solamente per generare nuovi blocchi. Il costo dei dispositivi e dell'elettricità necessari al mining sono la stake in questo algoritmo.
Un sistema che utilizza PoW potrebbe essere soggetto all'**attacco del 51%**. Questo attacco si concretizza nel caso in cui un utente o un gruppo di individui riesca a controllare la maggior parte della potenza di mining della rete. Nonostante sia teoricamente possibile l'attacco non è per nulla redditizio in quanto servirebbe una potenza di calcolo enorme per effettuarlo e farebbe abbandonare la rete compromessa agli altri utenti.

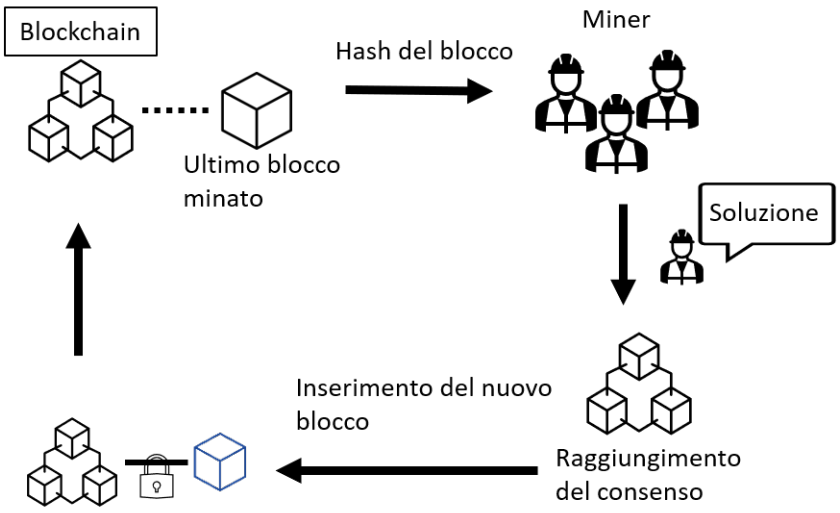

Figura 1.3: Schema del mining di nuovi blocchi

Proof of Stake

Questo algoritmo non comporta un uso massiccio di energia elettrica e non è necessario l'acquisto di hardware specializzato, per questi motivi si sta diffondendo in alcune blockchain importanti tra cui Ethereum. In questo caso, come posta in gioco per poter generare nuovi blocchi, è necessario mettere in *staking* una quota minima della criptovaluta della blockchain che utilizza la PoS.
L'algoritmo Proof Of Stake utilizza un processo di elezione pseudo-casuale per selezionare un nodo che agirà da validatore del blocco successivo, in base ad una combinazione di fattori che possono includere periodo di staking, randomizzazione e fondi di proprietà del nodo. L'inserimento di nuovi blocchi viene chiamato *forging* e come premio non vengono create nuove monete ma si distribuiscono le commissioni sulle transazioni. Tra tutti i blocchi candidati viene scelto quello selezionato dalla maggioranza dei possibili validatori e su cui essi hanno puntato una certa quantità di criptovaluta. Il **validatore** che è stato scelto dovrà occuparsi di verificare che tutte le transazioni all'interno del blocco siano valide, in seguito inserirà il blocco nella blockchain. Nel caso in cui un nodo dovesse validare alcune transazioni fraudolente, perderebbe tutta la posta in gioco messa inizialmente. Le diverse blockchain possono utilizzare metodi diversi per la scelta del prossimo blocco da aggiungere e del prossimo nodo validatore. Anche questo metodo di consenso è soggetto all'attacco del 51% ma più è alto il valore della criptovaluta e più diventa praticamente impossibile possederne una quantità così ampia.

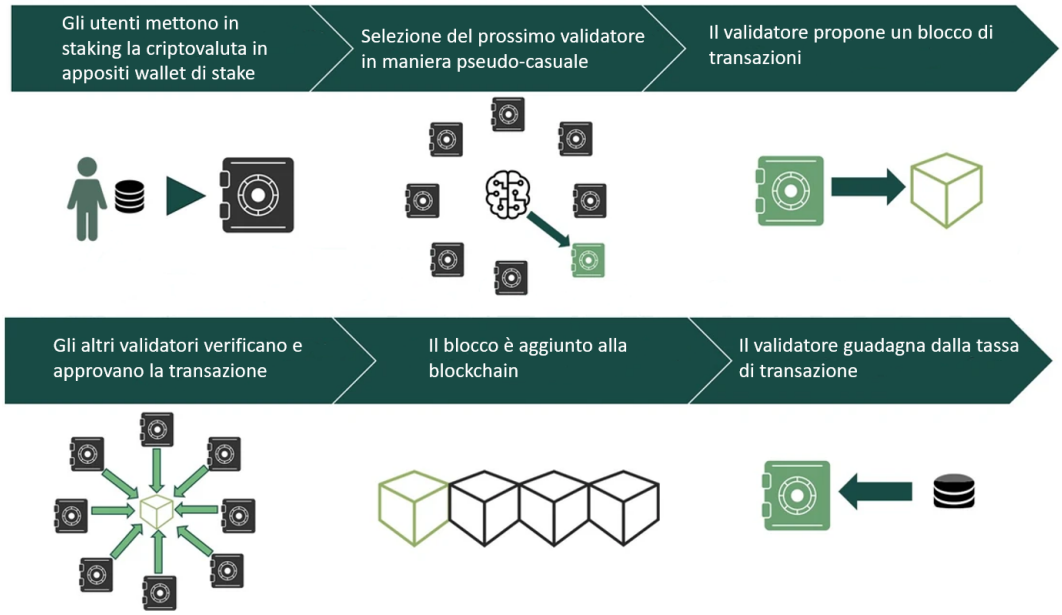

Figura 1.4: Schema riassuntivo della Proof of Stake (Fonte: SEBA Research)

Proof of Authority (PoA)

La Proof of Authority (PoA) è un algoritmo di consenso basato sulla reputazione che introduce una soluzione pratica ed efficiente per le reti blockchain (soprattutto quelle private). L'algoritmo di consenso PoA fa uso del valore delle identità. Ciò significa che i convalidatori dei blocchi mettono in stake la propria reputazione al posto delle monete virtuali. Di conseguenza, le blockchain PoA sono protette dai nodi di convalida che vengono selezionati arbitrariamente come entità affidabili. Il modello Proof of Authority si basa su un numero limitato di convalidatori, fattore che lo rende un sistema altamente scalabile. I blocchi e le transazioni sono verificati da partecipanti pre-approvati, che fungono da moderatori del sistema.
Il modello Proof of Authority consente alle imprese di mantenere la propria privacy e allo stesso tempo avvalersi dei vantaggi della tecnologia blockchain. Microsoft Azure è un esempio di compagnia che applica la PoA. Le condizioni possono cambiare al variare della blockchain ma l'algoritmo dipende solitamente da convalidatori che devono confermare la propria identità reale, la difficoltà nel diventare un convalidatore e uno standard per l'approvazione di nuovi convalidatori. I principali difetti della PoA sono la scarsa decentralizzazione e la conoscenza delle identità dei convalidatori che potrebbe portare alla manipolazione da parte di terzi.

1.1.2 Pregi e difetti della blokchain

Le principali caratteristiche della blockchain sono:

- **Distribuita**: i dati sono archiviati in tutti i nodi della rete e di conseguenza sono altamente resistenti ad attacchi o errori tecnici.
- **Immutabile**: è molto improbabile che i blocchi confermati vengano invertiti e annullati, quindi una volta che sono stati registrati nella blockchain, i dati sono estremamente difficili da rimuovere o modificare.
- **Sistema trustless**: in gran parte dei sistemi di pagamento tradizionali, le transazioni non dipendono solo dalle due parti coinvolte ma anche da un intermediario, nei sistemi blockchain invece le transazioni vengono verificate dai nodi della rete.

Tutte queste proprietà la rendono una tecnologia molto sicura, che riesce ad evitare la censura o la manipolazione dei dati da parte di entità centrali e permette ad ogni nodo di avere una copia di tutte le transazioni avvenute fino a quel momento.
Gli **svantaggi** principali sono: la possibilità di effettuare l'attacco del 51%, nonostante sia altamente improbabile che possa avvenire realmente; la difficoltà nel cambiare i dati, che rende necessari gli *hard fork*; l'inefficienza, soprattutto in blockchain che utilizzano la PoW e infine la dimensione del registro distribuito cresce molto nel tempo, portando all'eventuale perdita di nodi che non hanno spazio disponibile. La natura distribuita e il

modello cooperativo rendono robusto e sicuro il processo di validazione, ma presentano tempi non trascurabili, dovuti in gran parte al processo di validazione dei blocchi e alla sincronizzazione delle rete.

1.1.3 La blockchain Ethereum

Ethereum è una piattaforma decentralizzata del Web 3.0 per la creazione e pubblicazione peer-to-peer di contratti intelligenti (smart contracts) creati in un linguaggio di programmazione Turing-completo. L'idea dietro ad ethereum è stata concepita da Vitalik Buterin, è lui che ha lanciato la prima versione della piattaforma nel 2015 con l'aiuto di diversi co-fondatori. A differenza di Bitcoin, Ethereum opera utilizzando conti e saldi secondo le cosiddette transizioni di stato, che non si basano su output di transazione non spesi (unspent transaction outputs , UTXOs), ma sui saldi correnti (chiamati stati) di tutti i conti, oltre ad alcuni dati aggiuntivi.
La blockchain di ethereum permette anche ai programmatori di costruire e distribuire applicazioni decentralizzate (*dapps*) e queste vengono conservate nella blockchain insieme alla registrazione delle transazioni. Per poter girare sulla rete peer-to-peer, i contratti di Ethereum "pagano" l'utilizzo della sua potenza computazionale tramite un'unità di conto, detta Ether, che funge quindi sia da criptovaluta che da carburante. In altre parole, contrariamente a molte altre criptovalute, Ethereum non è solo un network per lo scambio di valore monetario, ma una rete per far eseguire contratti intelligenti scritti appositamente per svolgere alcune funzioni. Questi contratti possono essere utilizzati in maniera sicura per eseguire un vasto numero di operazioni: sistemi elettorali, registrazione di nomi di dominio, mercati finanziari, piattaforme di *crowdfunding*, proprietà intellettuale e molte altre applicazioni.
La blockchain Ethereum è nata utilizzando PoW come algoritmo di consenso ma è ora in una fase di transizione che porterà ad utilizzare la PoS per aumentare la scalabilità e l'efficienza della rete che è stata spesso criticata per le sue tasse troppo alte e la sua lentezza.

Smart contract

Uno smart contract è un contratto autoeseguibile con i termini dell'accordo tra acquirente e venditore direttamente scritti nel codice. Sono nati a metà degli Anni '70 e all'epoca servivano per gestire l'attivazione o la disattivazione di una licenza software in funzione di alcune condizioni. Lo smart contract deve fornire una serie di garanzie a tutte le parti coinvolte, per esempio che il codice con cui è stato scritto non possa essere modificato, che le fonti di dati che determinano le condizioni di applicazione siano certificate e affidabili e che le modalità di lettura e controllo di queste fonti sia a sua volta certificato.
Lo smart contract elabora in modo deterministico (con identici risultati a fronte di identiche condizioni) le informazioni che vengono raccolte. Gli smart contract, così come li

conosciamo oggi, hanno bisogno della blockchain per garantire quella fiducia nei rapporti tra le varie parti che non dipende più da una terza parte centralizzata, ma dai nodi della rete che si occupano di elaborare le transazioni. Ogni smart contract è verificabile in quanto il codice sorgente è accessibile a tutti.
Il **Gas** in Ethereum è un'unità di misura utilizzata per misurare il lavoro svolto dalla rete per effettuare transazioni o qualsiasi altra interazione. Ogni riga di codice degli smart contract richiede una certa quantità di gas per essere eseguita. Per far sì che un utente possa effettuare una transazione accettata e inclusa nella blockchain, è necessario che esso paghi una commissione affinché i minatori prendano la sua transazione e la includano in un blocco. Gli sviluppatori di Ethereum hanno deciso di assegnare valori costanti alle diverse operazioni che possono essere eseguite, in questo modo ogni attività sulla blockchain ha un valore di Gas stabilito, che non cambia e non viene alterato dall'aumento o dalla diminuzione del valore di Ether.
Sono presenti tre diversi fattori che influenzano il prezzo delle transazioni:

- **Unità di Gas**: è la quantità di Gas che può essere attribuita ad una specifica istruzione, ma non ha valore monetario.
- **Prezzo del Gas**: è il pagamento della commissione che effettuiamo per ciascuna unità di Gas. Rappresenza il prezzo che decidiamo di pagare per ogni unità, maggiore è questo valore e più velocemente i minatori prenderanno la transazione e la aggiungeranno in un blocco.
- **Limite di Gas**: è un valore che indica il numero massimo di unità di Gas che il mittente della transazione è disposto a fornire.

Il limite di Gas di una transazione standard è di circa 21000 unità di Gas, nel caso in cui sia impostato un valore più alto questo verrà rimborsato al mittente, nel caso in cui il valore impostato sia troppo basso tutto il Gas consumato sarà perso e la transazione sarà annullata. Ogni unità di Gas ha un prezzo in Gwei, dove un Gwei corrisponde a $1 * 10^{-9}$ ether.
Se la rete è congestionata per via delle numerose transazioni il prezzo da pagare si alza in quanto ogni blocco può consumare al più una quantità fissata di Gas. Utilizzando il Gas la blockchain Ethereum è in grado di difendersi da eventuali attacchi DOS o dal problema della terminazione. Nella figura seguente si può osservare un esempio di una transazione sulla blockchain Ethereum.

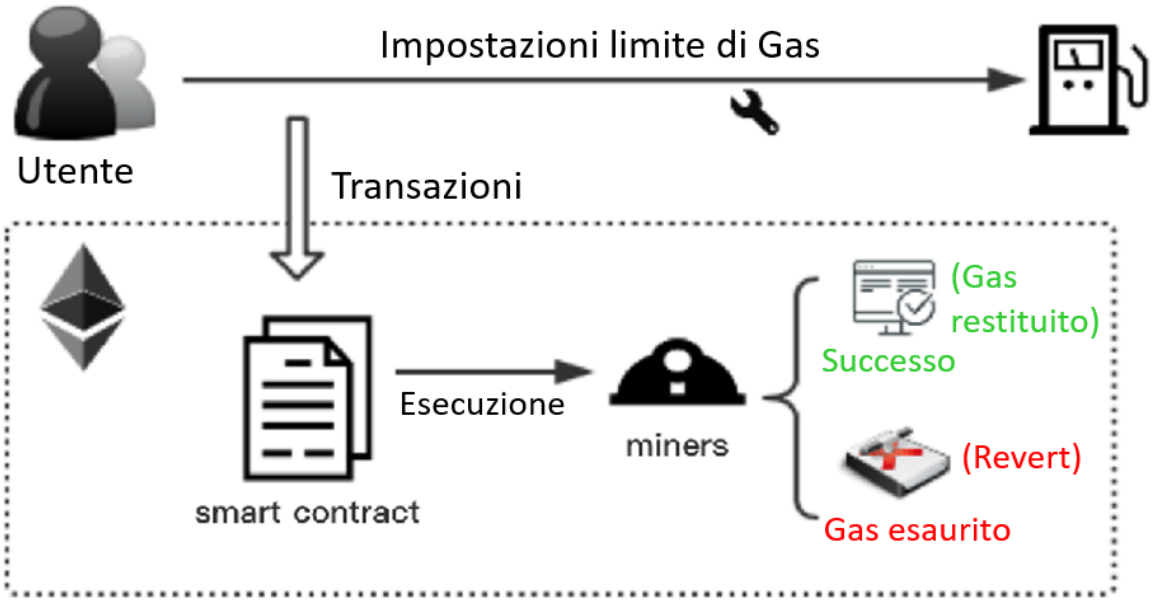

Figura 1.5: Esempio di transazione su Ethereum (Fonte: [2])

Capitolo 2

Metodi di contraffazione e sistemi per combatterli

2.1 La contraffazione

Nel Capitolo 1 è stata esposta la definizione di contraffazione e sono stati mostrati i dati delle perdite che essa comporta per le aziende, ora si entrerà più nel dettaglio esaminando i metodi utilizzati dai contraffattori. I costi della contraffazione non si circoscrivono alla sola perdita di ricavi dovuti all'aumento delle vendite dei prodotti contraffatti, ma essi derivano anche dalla perdita di distintività e di immagine legata alle esperienze negative dei consumatori con beni contraffatti che pensavano fossero originali.
Con riferimento al mondo della moda, sono tre i fenomeni contraffattivi di particolare rilevanza: quelli relativi ai marchi (1), ai prodotti (2) e alle commercializzazioni (3).

1. Il primo si realizza in presenza di un'imitazione del marchio che conduce a confusione sotto un duplice aspetto:

   (a) in alcuni casi il marchio viene riprodotto fedelmente e apposto su prodotti simili o identici a quelli originali;

   (b) in altri casi il marchio è soltanto simile, ma tale da richiamare alla mente del consumatore quello originale.

   L'utilizzo di marchi molto simili a quelli più conosciuti è considerata una delle pratiche più pericolose nelle relazioni commerciali perché può confondere il consumatore che deve acquistare il prodotto, infatti il cliente non sempre ha un ricordo perfetto di come sia fatto il relativo marchio.

Figura 2.1: Esempio di logo contraffatto

2. Nel secondo caso vengono imitate le linee di un abito che, magari insieme all'utilizzo di un marchio simile a quello dell'azienda che ha disegnato il vestito originale, può ingannare il consumatore che lo acquista credendo di comprare il prodotto di marca originale.

3. Infine si presenta il caso in cui un prodotto viene creato e commercializzato utilizzando un marchio identico a quello delle grandi case di moda senza che tuttavia queste lo producano. Inoltre si può considerare il fenomeno delle **importazioni parallele**, ossia quel fenomeno attraverso cui vengono commercializzati in una determinata area geografica prodotti originali che però non erano destinati a tale mercato, quindi il prodotto è originale in tutto e per tutto, ma il titolare dei diritti di sfruttamento economico sullo stesso non ne ha autorizzato la vendita.

L'evoluzione e la dilagazione esponenziale del fenomeno della contraffazione è dovuta anche ai massimi utilizzi nel nostro periodo storico di piattaforme di vendita online. Si è infatti creato un ambiente di vendite sofisticato con siti costruiti in maniera impeccabile ed alcuni che offrono la possibilità di pagare in modi conosciuti e sicuri (p.e. Paypal). Il web garantisce ai contraffattori (o a chi commercializza prodotti falsi) anonimità e facilità di aprire e chiudere gli e-store. La rete di falsi online è difficile da smantellare, considerato che la transnazionalità degli illeciti porta con sé una serie di limiti di giurisdizione e di competenza.
Esistono due tipologie di consumo dei beni contraffatti: *ingannevole* e *non ingannevole*. Fanno parte della prima categoria tutti gli acquisti in cui il consumatore non è in grado di distinguere la non autenticità del prodotto che sta per acquistare. Invece nel secondo caso, che riguarda prevalentemente il mercato dei beni del lusso, il consumatore è in grado di distinguere il prodotto originale da quello contraffatto ma per questioni di prezzo o comodità lo acquista.

2.2 Metodi di contraffazione

È importante focalizzare l'attenzione su quali siano le tattiche adottate dai contraffattori e di quali strategie si servano per raggiungere il loro obiettivo. I contraffattori, come in qualsiasi attività a scopo di lucro, hanno come obiettivo primario quello di massimizzare il profitto, quindi dovendo mettere in commercio un prodotto non autentico, devono renderlo quanto più possibile simile all'originale per dargli credibilità.
Da un'analisi delle modalità attraverso le quali si sviluppa il fenomeno della contraffazione sono emersi quattro canali principali, essi sono: imitazione (o *knockoff*), *reverse engineering*, *third shift* e *outsourcing issue* [3].

1. **Nel knockoff** il prodotto assomiglia all'originale ed il consumatore è consapevole della falsità ma paga un prezzo inferiore per l'acquisto. Questo determina una perdita indiretta da parte dell'azienda legittima.

2. Si parla di **reverse engineering** quando la merce illecita viene spacciata come originale, l'utente non sa che si tratta di un falso e viene quindi truffato[1].

3. **Third shift** è quando una compagnia ufficiale per un certo periodo ha firmato un contratto con un produttore e, al termine del contratto, il produttore invece di liquidare le macchine usate per la produzione della merce continua a produrre e vendere.

4. **Outsourcing issue** si ha quando il produttore non rispetta gli standard di produzione prestabiliti.

Inoltre sono state individuate quattro macro strategie entro le quali può collocarsi il *modus operandi* dei contraffattori.

- La strategia di **estrazione**: sono prodotti che vengono dismessi in un paese e venduti in un altro, nel quale vengono adattati e rivenduti come se fossero nuovi.

- La strategia di **produzione**: si basa sull'anticipazione del lancio del prodotto sul mercato, si servono di questo metodo aziende che hanno grossi colossi come rivali e che quindi cercano di diffondere prodotti falsi simili agli originali in anticipo. In alcuni casi quando le grandi aziende spostano la produzione in paesi più poveri succede che gli stessi lavoratori che di giorno lavorano in fabbriche "ufficiali", nel tempo libero sfruttino le loro conoscenze per produrre i falsi per la criminalità.

- La strategia di **distribuzione**: per sfuggire ai controlli spesso la merce viene raggruppata in piccoli lotti in modo da essere difficilmente tracciabile. Un'altra tecnica

[1] L'imitazione servile è un atto di concorrenza sleale che consiste nell'imitazione fedele e pedissequa dei prodotti di un concorrente, tale da creare confusione nel pubblico sulla provenienza degli stessi.

consiste nell'oscurare il marchio falsificato finché il prodotto non arriva al consumatore, in questo modo durante il trasporto non si sospetta di nulla ed è possibile che vengano vendute merci originali con degli accessori falsi.

- La strategia dell'**infiltrazione**: consente ai prodotti contraffatti di introdursi nel flusso dei prodotti originali. Alcuni prodotti necessitano infatti di un packaging e di istruzioni nella lingua del paese in cui saranno venduti, perciò nel momento in cui avviene il rimpacchettamento vengono inseriti anche prodotti falsi.

2.3 Stato dell'arte dei sistemi anticontraffazione

Esistono alcune soluzioni sviluppate nel corso degli anni per combattere la contraffazione ma nessuna è pienamente efficace. Negli ultimi anni, grazie alla diffusione della blockchain, sono stati progettati e ipotizzati sistemi nei quali si combina l'utilizzo della blockchain con il tracciamento tramite RFID o altri sensori. La blockchain sembra essere la soluzione ideale per problemi di tracciamento e certificazione dell'originalità dei prodotti perciò alcune grandi aziende stanno lavorando per produrre un loro sistema anticontraffazione che utilizzi questa tecnologia.

2.3.1 Sistemi anticontraffazione canonici

I metodi anti-contraffazione possono essere oggetto di differenti tipi di classificazione in base alla considerazione di aspetti differenti. Si può scegliere di adottare una classificazione che si basa essenzialmente sull'aspetto tecnologico distintivo considerato predominante, che consente un'identificazione più semplice di ciascun metodo anti-contraffazione.

- Tecnologie **elettroniche**: includono RFId (passivi, attivi), NFC, sigilli elettronici, banda magnetica e chip a contatto.
- Tecnologie di **marchiatura**: possono essere di tipo visibile e invisibile e comprendono le tipologie di banda a memoria ottica, codici a lettura ottica (p.e. barcode), ologrammi, inchiostri, Copy Detection Patterns.
- Tecnologie **chimico-fisiche**: includono tecniche basate su codifica chimica, traccianti micro e nanotecnologie.
- Tecnologie **meccaniche**: comprendono etichette, incisioni laser e sigilli.

I prodotti più utilizzati dalle aziende sono le etichette di vario genere: **ultraresistenti**, utili per facilitare la rintracciabilità a livello internazionale lungo tutta la filiera; le etichette **Void**, che depositano sul prodotto una parte della propria colorazione contrassegnando il prodotto protetto con la scritta standard "Void" (in inglese: violato); le

etichette RFID.
La maggior parte delle aziende utilizza come soluzione una combinazione di NFC e QR Code, il primo serve per garantire l'autenticità e il secondo per poter osservare le informazioni relative al prodotto in un sito web. Le maggiori vulnerabilità di questi sistemi non risiedono nelle tecnologie di marcatura ed identificazione ma nel fatto che siano basati su un modello client-server e siano centralizzati. I server e i database utilizzati per la gestione del servizio possono essere hackerati o attaccati per impedirne il corretto funzionamento e i dati potrebbero essere alterati.
Per poter risolvere questi problemi si può ricorrere all'utilizzo di un sistema basato sulla blockchain che impedirebbere l'alterazione dei dati, ogni nodo della rete avrebbe una copia di essi e tutte le operazioni sarebbero trasparenti.

2.3.2 Sistemi anticontraffazione che utilizzano la blockchain

In letteratura sono già stati implementati o sono in via di implementazione alcuni sistemi anticontraffazione che utilizzano la blockchain. La prima soluzione esaminata è il **dNAS** [4], un sistema che propone di aggiungere la blockchain al **NAS**, un progetto già esistente per l'anticontraffazione nell'industria del vino. L'obiettivo è quello di rendere decentralizzato il vecchio sistema per facilitare il reperimento, la verifica e la gestione di dati affidabili sulla provenienza dei prodotti e rafforzare la qualità dell'anticontraffazione e della tracciabilità nell'industria del vino.
Il NAS è basato sull'utilizzo della tecnologia NFC, un'architettura di microservizi basati sul cloud e una struttura di archiviazione centralizzata, ospitata dai produttori di vino. Questa architettura centralizzata presenta vari problemi: il server è sottoposto a pesanti processi di elaborazione dei dati; i sistemi anticontraffazione e tracciabilità come il NAS, fanno affidamento su un'autorità centralizzata e questo porta ad un unico punto di elaborazione, memorizzazione ed errore in caso di guasti o attacchi [5]. Con l'aggiunta di una blockchain al sistema si avrebbero molti benefici in quanto ogni transazione sui dati dei prodotti richiederebbe una validazione on-chain e off-chain e la disponibilità e resistenza dei dati migliorerebbe in quanto ogni nodo possederebbe una copia dei dati. Si avrebbe però una gestione più complicata, una scalabilità che dipende dal livello di decentralizzazione e gli altri problemi conosciuti della blockchain.
Per concludere, il dNAS è costituito da vari componenti tra cui la blockchain, gli smart contract scritti per interagire con il sistema, l'interfaccia utilizzata dagli utenti e IPFS per la memorizzazione dei dati. Nella tabella a seguire verrà mostrato un confronto tra l'architettura decentralizzata del dNAS e quella centralizzata del NAS.

Architettura decentralizzata con blockchain	Architettura centralizzata
Dati aggiunti solamente quando si raggiunge il consenso	Dati aggiunti tramite gli amministratori senza alcun consenso raggiunto
Si possono solo inserire nuovi dati, i vecchi sono immutabili	Non ci sono restrizioni sulle modifiche ai dati
Distribuita per come è costruita la blockchain	Singolo punto di errore
Decentralizzata per come è costruita la blockchain	Singolo punto di controllo
Struttura peer–to-peer con istanze cloud	Architettura client-server
Verifiche crittografiche	Server esegue azioni per conto degli utenti
Resilienza e disponibilità aumentano col numero dei nodi	Backup e piani di emergenza implementati manualmente
Autenticazione e autorizzazione crittografica	Crittografia implementata separatamente come funzione aggiuntiva

Figura 2.2: Confronto tra architettura decentralizzata e centralizzata

VeChain è un sistema già attivo che offre soluzioni per le aziende interessate al tracciamento e la verifica dei dati dei propri prodotti [6]. Utilizza una propria blockchain insieme agli smart tag (NFC, RFID e QR Code) per tracciare la gestione del ciclo di vita dei prodotti e permette agli utenti di verificare e controllare le informazioni relative ad un prodotto tramite la VeChain Pro app o tramite l'SDK fornito agli sviluppatori.
Aura [7] è un progetto nato per mostrare ai clienti la storia del prodotto e provare la sua autenticità, tracciandolo dalla fonte al venditore. I fondatori del progetto sono le aziende Louis Vuitton, Prada e Cartier e molte altre possono entrare a farne parte. Dall'utilizzo del sistema ne traggono vantaggio anche i venditori (brand) in quanto assicurano che i prodotti acquistati dagli utenti siano fatti secondo gli standard stabiliti dalle aziende generando fiducia nei clienti senza il bisogno di intermediari e allo stesso tempo proteggendosi dalla contraffazione. Il processo si divide in varie fasi:

- *Produzione*: creare un ID digitale unico per ogni prodotto per permettere la tracciabilità.
- *Distribuzione*: mantenere un registro di ogni transazione nel certificato del prodotto.
- *Acquisto*: consegnare un certificato di autenticità garantendo la genuinità del prodotto e dando accesso alle informazioni riguardanti la sua produzione.
- *Garanzia*: tenere traccia di interventi sul prodotto (p.e. manutenzione e riparazioni).

- *Rivendita*: garanzia dell'autenticità del prodotto con il certificato e utilizzo di un algoritmo per rilevare i prodotti contraffatti.

Anche altre grandi aziende, tra cui l'IBM, lavorano già da qualche anno con la blockchain per sviluppare soluzioni innovative che sfruttino tutto il potenziale di questa tecnologia. Inoltre si sta valutando e utilizzando la blockchain in collaborazione con l'Internet of Things. Questo genere di oggetti, sempre più diffusi nella vita quotidiana di milioni di persone, raccolgono grandissime quantità di dati. Potrebbero essere usati dei piccoli oggetti per tracciare costantemente i prodotti e avere la posizione o altre informazioni aggiornate in tempo reale sulla blockchain. Bisogna fare attenzione alle numerose vulnerabilità che spesso questi oggetti possiedono, in quanto per risparmiare sui costi di produzione e sviluppo si utilizza hardware poco costoso e si effettuano scelte implementative sbagliate.

2.3.3 Problemi RFID

Il tag RFID è costituito da un microchip che contiene dati in memoria, un'antenna e un supporto fisico che mantiene uniti il chip e l'antenna. Tutti i tag RFID, non protetti con tecniche di crittografia, sono soggetti ad attacchi come la clonazione e la modifica o la rimozione dei dati salvati all'interno della memoria. In generale i tag attivi si riescono a proteggere in maniera più semplice rispetto a quelli passivi in quanto presentano piccole batterie che consentono di effettuare operazioni più complesse.
Nel sistema proposto, nel caso in cui venisse modificato o eliminato il codice univoco del prodotto al quale il tag è collegato, tutte le informazioni relative al prodotto non sarebbero più direttamente collegate ad esso e non se ne potrebbe più dimostrare l'autenticità.
Se un tag RFID venisse clonato ed associato ad un prodotto falso, ci sarebbero due prodotti considerati validi sul mercato senza sapere quale dei due sia effettivamente l'originale. Per il riconoscimento sarebbero necessarie ulteriori analisi sui prodotti per ottenere informazioni sulla provenienza.
Se un tag RFID di un prodotto certificato subisse una clonazione e in seguito fosse modificato, la merce contraffatta verrebbe scambiata per quella autentica non permettendo all'originale di essere riconosciuta in quanto il codice posseduto sarebbe invalido . I metodi per evitare la clonazione o l'alterazione dei dati presenti in un tag RFID sono vari, tra cui l'utilizzo di una firma digitale, di chiavi simmetriche o di chiavi derivate. Nel caso delle *chiavi derivate* si genera una chiave differente per ogni coppia di tag e lettore RFID che comunicano. Questa chiave viene generata a partire dal numero di serie del tag e da un segreto condiviso che è contenuto nel firmware di entrambi. A questo punto il reader genererà la chiave leggendo il codice univoco dal tag mentre il tag, conoscendo il suo numero di serie, sarà già a conoscenza della chiave e quindi non sarà necessario comunicarla, aumentando la sicurezza della procedura.

Capitolo 3

Analisi del sistema proposto e confronto con quelli già esistenti

L'obiettivo del sistema proposto è quello di impedire l'utilizzo di certi metodi di contraffazione e di aiutare nel processo di tracciamento e quindi verificabilità dei prodotti provenienti dalle aziende del mondo della moda. Nel nostro sistema ogni prodotto viene etichettato utilizzando la tecnologia RFID, in seguito ogni volta che il prodotto viene trasferito o scambiato si effettuano transazioni tramite gli smart contract, in questa maniera tutti i dati rimangono salvati sulla blockchain ed in caso di necessità sono disponibili. Il sistema è stato progettato fin dalla sua nascita per lavorare con dei prodotti creati da fabbriche autorizzate che lavorano per brand importanti nel mondo della moda. Principalmente ci si riferirà ad aziende che vendono prodotti di alta qualità e di lusso per via dei costi in termini di tempo e risorse per adeguare i sistemi odierni ad un'eventuale interazione con la blockchain.
Per evitare tentativi di inserimento di merci contraffate nel ciclo produttivo o altre alterazioni del flusso di trasporto regolare dei prodotti, si tiene traccia dei dati sulla blockchain dalla creazione nella fabbrica fino alla vendita in negozio.

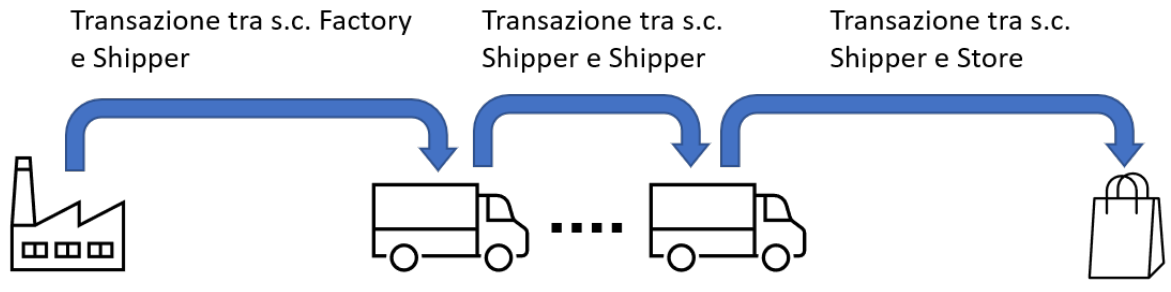

Figura 3.1: Ciclo di vita del prodotto con registrazione delle transazioni sulla blockchain

Grazie all'utilizzo della blockchain è possibile evitare la manipolazione dei dati una volta che sono stati registrati su di essa e allo stesso tempo permette all'utente finale e ai vari intermediari di verificare i dati relativi al prodotto e ai corrieri che lo hanno trasportato. Nel sistema proposto ogni prodotto è identificato da un codice univoco salvato in un'etichetta o in un tag tramite RFID. Questa tecnologia è utilizzata per l'identificazione e la memorizzazione di informazioni tramite radiofrequenza. In questa maniera ogni prodotto che esce da una fabbrica autorizzata potrà essere sempre identificato tramite la lettura del codice univoco che gli è stato assegnato al momento della creazione. Il sistema illustrato si basa sull'univocità degli RFID assegnati ai prodotti, infatti in ogni momento avendo il codice si possono ottenere le informazioni sul prodotto da uno dei vari attori coinvolti. In un chip RFID si possono salvare diversi dati in base a ciò che è necessario, nel nostro caso è stato ipotizzato che venisse salvato semplicemente un codice univoco che servisse ad identificare ogni prodotto.
I principali problemi legati all'utilizzo della tecnologia RFID sono stati esposti nella sezione 2.3.3.

3.1 Utilizzo del progetto contro le strategie di contraffazione

Ora verrà analizzato nel dettaglio come il progetto sviluppato può essere in grado di combattere alcuni dei metodi utilizzati per la contraffazione dei prodotti nel mondo della moda. In generale si può affermare che la tecnologia blockchain sia ideale per quanto riguarda tutti i processi di tracciamento, contabilità e certificazione per via delle sue caratteristiche. Si osservano, elencando le sue proprietà, i motivi per cui viene considerata una tecnologia ideale per questo genere di applicazioni:

- Decentralizzazione: essendo decentralizzata si evita di avere il cosiddetto *single point of failure* e si riescono ad aggiungere nodi alla rete in maniera scalabile.
- Trasparenza: ogni persona può accedere alla blockchain e osservare le transazioni avvenute e dati salvati su di essa.
- Sicurezza e Immutabilità: grazie all'utilizzo di crittografia e di vari algoritmi di consenso la blockchain è resistente anche se alcuni nodi sono malevoli.

Strategia di estrazione

Si analizza per primo il tentativo da parte dei contraffattori di rivendere prodotti ritirati dal mercato per la presenza di difetti o la comparsa di altri problemi. Per avere successo, devono cercare di modificare i prodotti in maniera tale che essi appaiano come nuovi.

Figura 3.2: Strategia di estrazione senza tracciamento dei prodotti

Si assuma che ogni prodotto di una certa marca sia tracciato con il metodo proposto. Una volta che viene ritirato dal mercato per un qualche motivo, si può modificare una variabile all'interno del prodotto per indicare il cambiamento di stato, in questa maniera il prodotto sarà marcato come "da ritirare".
Ora verrà ipotizzata una situazione simile a quella rappresentata in figura 3.2.

1. Il contraffattore prende il controllo del prodotto ritirato, che però ha un RFID ed è stato tracciato con il nostro sistema. Sulla blockchain nei dati relativi a quel prodotto compare anche la dicitura "da ritirare", visibile quindi a chiunque osservi lo storico della blockchain.

2. Il prodotto viene modificato per farlo apparire come nuovo e rimetterlo sul mercato.

3. Il prodotto viene rimesso in vendita, quindi o viene spedito tramite corrieri o venduto direttamente ad una persona.

4. A questo punto, nel caso in cui i corrieri dovessero ritirare il prodotto si accorgerebbero leggendo i dati del prodotto sulla blockchain che è stato marcato come "da ritirare" e potrebbero segnalarlo all'azienda che ha prodotto il capo d'abbigliamento. Anche nel caso in cui il cliente dovesse comprare il prodotto direttamente dal contraffattore si accorgerebbe dell'irregolarità consultando i dati sulla blockchain.

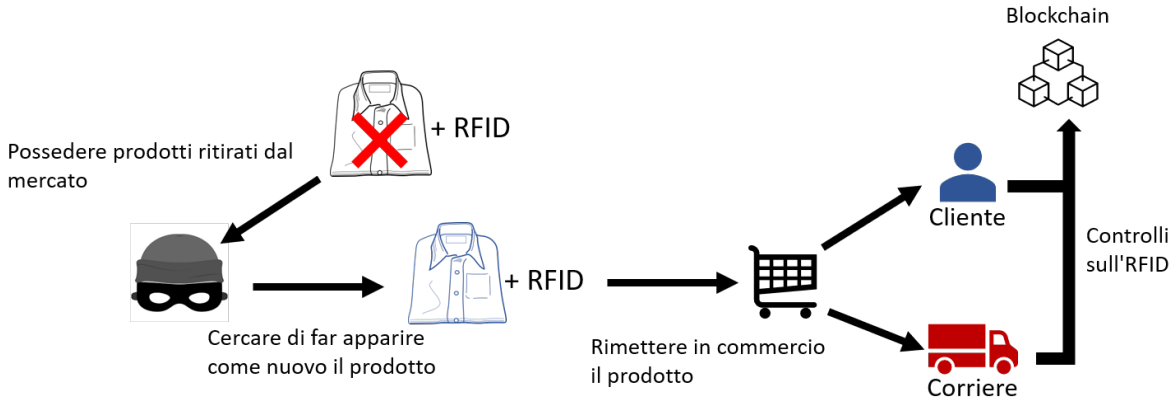

Figura 3.3: Strategia di estrazione con tracciamento prodotti

Chi dovrebbe effettuare la segnalazione di un eventuale difetto? Il negozio sembra essere la scelta migliore, in questa maniera controlla se realmente il prodotto è da ritirare e procede con la segnalazione e il rimborso al cliente. Il sistema ideato non si occupa della fase conclusiva di vita del prodotto, o di eventuali trasferimenti di oggetti da ritirare dal mercato, in ogni caso utilizzando la blockchain sarebbe possibile ampliarlo per occuparsi anche di questi aspetti in maniera del tutto analoga a quelli considerati.
L'unica debolezza riscontrabile in questo caso è la possibilità che un negoziante malevolo non segnali sulla blockchain un prodotto da ritirare, per poi rivenderlo a prezzo pieno. A questo punto nella blockchain i dati non sarebbero aggiornati e il prodotto verrebbe considerato come 'valido'. Purtroppo debolezze di questo tipo non sono risolvibili in quanto non dipendono dalla tecnologia o dalla blockchain ma dai dati inseriti o non inseriti volutamente.

Strategia di Produzione

Non è possibile impedire questo tipo di contraffazione con il sistema proposto ma si può assicurare ad eventuali clienti la possibilità di verificare che le merci acquistate siano originali grazie alla blockchain. Se un brand ha deciso di utilizzare una tecnologia di questo tipo ogni prodotto che verrà acquistato di quella determinata marca sarà rintracciabile sulla blockchain perchè seguirà i passi elencati all'inizio di questo capitolo. Un utente potrà quindi leggere l'RFID tramite un lettore e cercando sulla blockchain, con un'app fornita dal negozio, potrà essere sicuro dell'originalità.

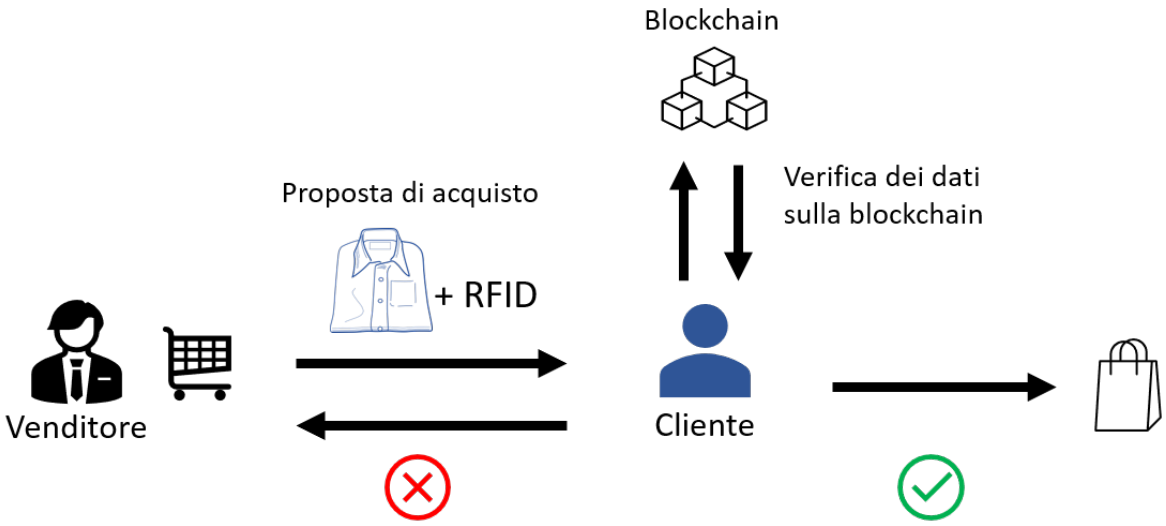

Figura 3.4: Verifica del prodotto prima dell'acquisto

Strategia di Distribuzione

La strategia di distribuzione cerca di eludere i controlli effettuati nascondendo il marchio durante tutto il trasporto oppure raggruppando in piccoli lotti per rendere più difficile il tracciamento. In entrambi i casi il nostro sistema fornisce una soluzione per risolvere i problemi. Essendo tutti taggati con la tecnologia RFID, i prodotti di un'azienda sono in ogni momento identificabili e si può cercare lo storico delle transazioni sulla blockchain utilizzata dall'azienda.
Nel **primo caso** anche se il marchio viene coperto per evitare controlli più restringenti sui prodotti di qualità, se la merce non presenta un RFID o non è presente in nessuna transazione sulla blockchain significa che il prodotto non proviene da una fabbrica autorizzata e di conseguenza non è originale.
Il **secondo caso** è analogo al primo in quanto una volta arrivata la merce ad un negozio, virtuale o fisico, il cliente prima dell'acquisto potrà verificare l'esistenza dell'RFID e i dati relativi alla vita del prodotto fino a quel punto. Una vulnerabilità causata dal cliente è la possibilità che esso si fidi dell'RFID comunicato dal venditore senza controllare che appartenga effettivamente al prodotto tramite una scansione autonoma. Nel caso in cui il venditore illegittimo fosse riuscito in qualche maniera a leggere un RFID di un prodotto autentico, potrebbe comunicarlo per mostrare l'apparente originalità di un prodotto falso. Utilizzando RFID sicuri questo attacco non dovrebbe essere possibile in quanto i dati del chip sono cifrati e quindi impossibili da leggere per individui non autorizzati.
Rimane però la possibilità che i clienti, per velocizzare le operazioni o perchè non a co-

noscenza dell'utilizzo da parte dell'azienda di un sistema di questo tipo, non effettuino i controlli sulla blockchain.

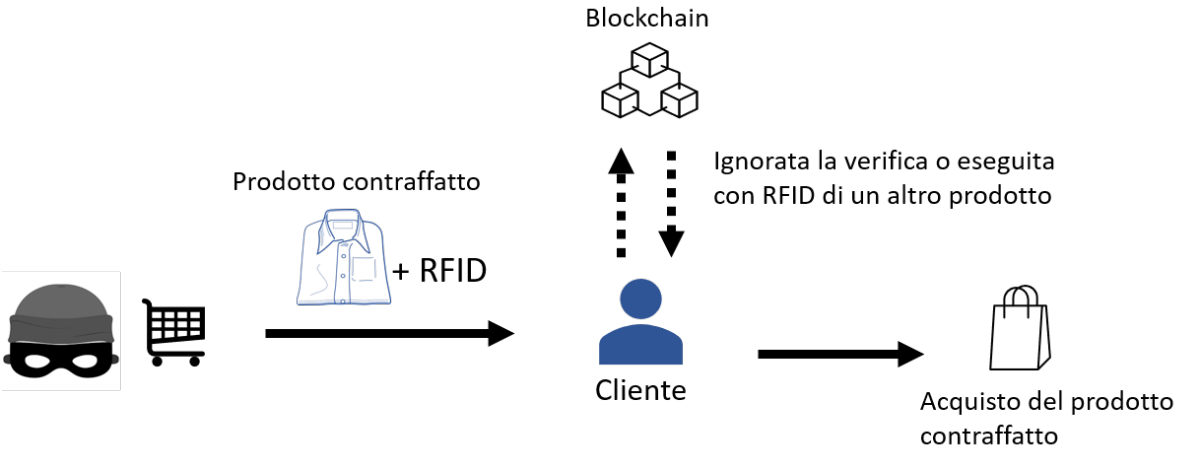

Figura 3.5: Possibile debolezza nel momento della vendita

Strategia dell'infiltrazione

L'ultima strategia esaminata è quella in cui prodotti autorizzati e quelli contraffatti vengono impacchettati insieme, questo solitamente accade nei processi di rimpacchettamento per il cambio della lingua delle etichette o del packaging.
Per come è stato scritto il sistema, eventuali inserimenti di prodotti falsi sono impediti nel momento in cui il prodotto originale esce dalla fabbrica. Infatti per ritirare i prodotti i corrieri devono fornire l'RFID del singolo elemento, avviene perciò un controllo molto fine. In ogni caso, anche se un prodotto viene aggiunto in qualche maniera tra quelli consegnati da un corriere autorizzato, ci si accorgerà dell'irregolarità una volta che i prodotti saranno consegnati al negozio, dove verranno di nuovo verificati i codici univoci presenti negli RFID.
Se per caso i prodotti contraffatti venissero aggiunti direttamente nel magazzino di un negozio, a questo punto sarebbe presente la vulnerabilità mostrata in figura 3.5.

Third shift e outsourcing issue

Per concludere si analizzano due casi interessanti che si discostano in un certo senso da quelli precedenti. Nel **third shift** il manifacturer continua a produrre capi d'abbigliamento di una certa marca anche dopo la scadenza del contratto con una compagnia. Conseguentemente viene considerata contraffazione in quanto questa fabbrica non è più autorizzata a produrre i capi d'abbigliamento per un marchio.

Come spiegato nella sezione 4.3.2, il nostro sistema prevede l'utilizzo di un oracolo per la creazione dei codici univoci RFID da inserire nei prodotti. Il codice JavaScript può essere modificato solamente da alcuni tecnici con determinati privilegi, ed è in ascolto sul server aziendale che a sua volta seguirà alcune politiche di sicurezza per evitare intrusioni e alterazioni dei dati. Per questo motivo una volta che il contratto con una determinata fabbrica è finito basterà impedire allo smart contract della fabbrica di contattare l'oracolo, inserendo l'indirizzo in una lista nera direttamente sull'oracolo on-chain oppure distruggendo il contratto con un'opportuna funzione.

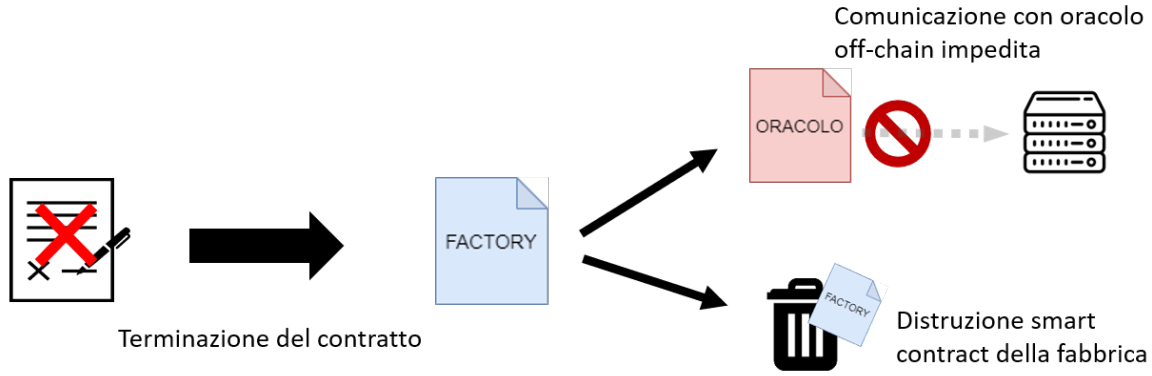

Figura 3.6: Third Shift con utilizzo del sistema

Il **problema dell'outsourcing** si presenta per il cliente e per il marchio originale che potrebbe subire un danno alla reputazione. L'outsourcing è una pratica comunemente utilizzata da molte aziende e consiste nell'affidarsi ad altre imprese per lo svolgimento di alcune fasi del processo produttivo. La situazione indesiderata sorge quando queste aziende terze non rispettano gli **standard di produzione** e qualità imposti dai proprietari del marchio. Tracciando con la blockchain tutte le informazioni relative ai prodotti si possono osservare anche le materie prime utilizzate per la creazione di un determinato capo d'abbigliamento. Nel caso in cui ci si accorga dell'utilizzo di materiali scadenti o differenti da quelli prestabiliti si può agire sospendendo la collaborazione.
È possibile che all'interno della fabbrica vengano però inseriti nei dati i materiali corretti senza poi utilizzarli realmente. Per evitare queste operazioni sarebbe necessario un controllo da parte dell'azienda che rappresenta il marchio su tutti i prodotti utilizzati dai manifacturer. In alternativa si può utilizzare la certificazione di un ente esterno che si occupa di controllare i materiali di produzione delle aziende. Il sistema senza questi controlli può essere vulnerabile all'inserimento di dati falsi sulla tipologia dei materiali utilizzati per creare un prodotto.

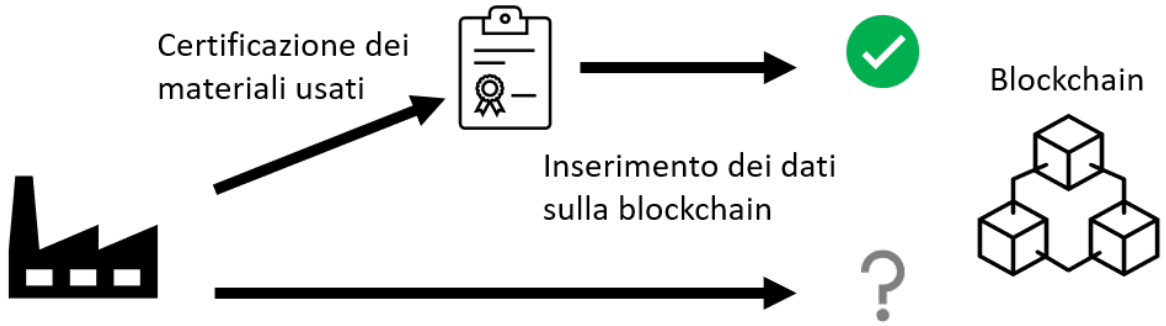

Figura 3.7: Outsourcing issue gestito con blockchain

In riferimento alla figura 3.7, se i dati vengono inseriti sulla blockchain dopo aver ottenuto una certificazione si può essere certi della loro veridicità, in caso contrario non è assicurato che i dati siano reali. Si ipotizza che nel nostro sistema vengano utilizzati RFID sicuri e robusti che non siano soggetti a certi tipi di attacchi anche se il sistema sarebbe resistente a qualche attacco dovuto ad RFID vulnerabili.

Possibile soluzione a RFID vulnerabili

Il terzo attacco illustrato nella sezione 2.3.3 può essere identificato se si agisce con delle analisi on-chain. Si potrebbe eseguire una scansione delle transazioni cercando l'eventuale esistenza di due prodotti con lo stesso RFID ma con percorsi che presentano una divergenza nella catena dei corrieri o che sono arrivati entrambi al negozio. In questo caso si viene dunque a conoscenza della clonazione ma non si riesce a capire quale dei prodotti arrivati è quello originale, se non con analisi approfondite sul prodotto effettuate da esperti fidati o dalla fabbrica di produzione autorizzata.

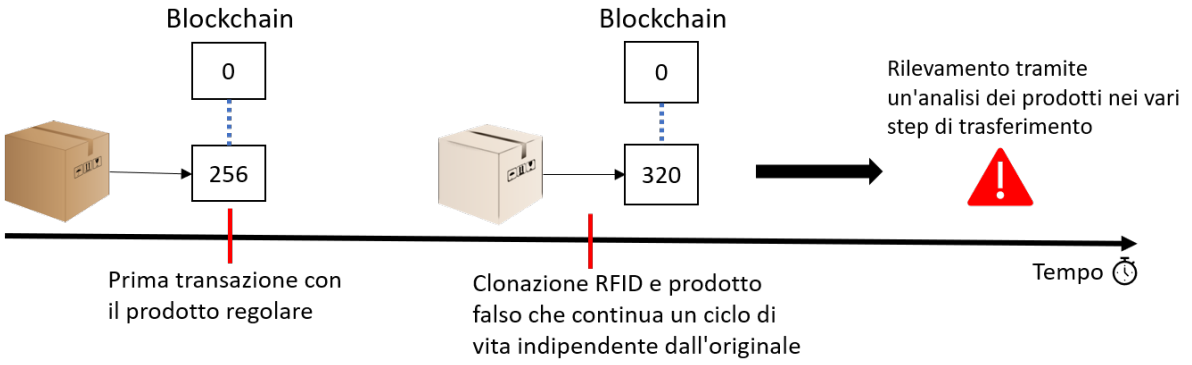

Figura 3.8: Rilevamento clonazione RFID tramite analisi transazioni

3.2 Vantaggi e debolezze del sistema

A questo punto verranno analizzate alcune debolezze del progetto che non dipendono dalla blockchain ma nella maggior parte dei casi da comportamenti degli utilizzatori del sistema.

- Il sistema non copre ogni aspetto possibile del trasferimento della merce, per esempio nel caso in cui siano da ritirare dal mercato i prodotti sarebbe da implementare un ulteriore procedura per modificare il loro stato sulla blockchain.
- Se il cliente non si preoccupa di verificare sulla blockchain i dati del prodotto potrebbe acquistare merce contraffatta.
- Nel caso in cui sia presente un venditore malevolo che ha acquistato la merce dalla fabbrica potrebbe contraffare i prodotti per avere un doppio guadagno nel caso in cui i clienti non controllino l'RFID.
- I dati inseriti sulla blockchain sono immutabili ma questo non garantisce che essi vengano inseriti in maniera corretta o che siano veritieri.

Inoltre ci sono complicazioni derivanti dalla progettazione e implementazione di questo sistema su larga scala e i costi iniziali sarebbero elevati. Per agevolare l'utilizzo del sistema e la verifica dei prodotti è necessaria la creazione di un'applicazione per i clienti. Se possibile sarebbe utile integrare nell'app un lettore per gli RFID per poter verificare autonomamente i dati senza doversi fidare del venditore. Al livello attuale della tecnologia questo non è possibile ed è necessario un lettore apposito, quindi nel sistema si potrebbe utilizzare al posto di un normale tag RFID, un chip NFC. Questa tecnologia può essere integrata negli smartphone e permette la cifratura dei dati, caratteristica necessaria nel sistema per impedire la clonazione ed eventuali modifiche al codice univoco dei prodotti.
I **benefici** che il sistema può portare derivano dall'utilizzo della blockchain e dalle sue proprietà. Permetterebbe ad un marchio di tracciare in maniera esaustiva i propri prodotti e combatterebbe alcune tra le tecniche di contraffazione più utilizzate; aumenterebbe il grado di fiducia dei clienti nei confronti dei prodotti acquistati, in quanto sarebbero in grado di verificare loro stessi tutti i dati; semplificherebbe il salvataggio e la gestione dei dati nei database per l'analisi degli acquisti e dei clienti in quanto i dati più significativi e importanti per ogni prodotto sarebbero salvati sulla blockchain.

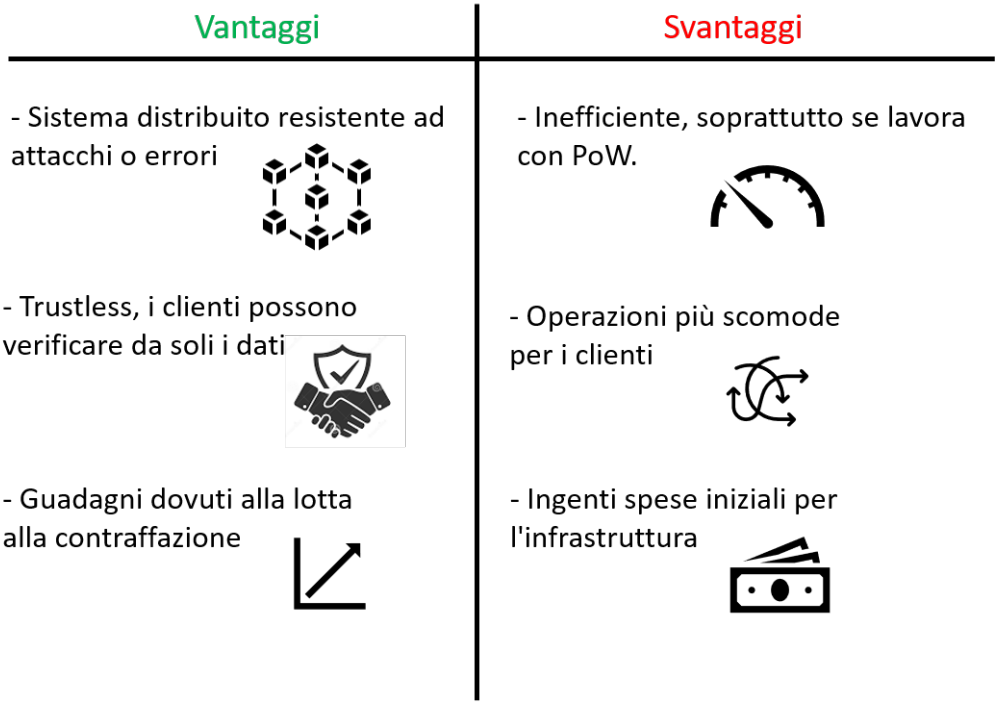

Figura 3.9: Pro e contro del sistema

3.2.1 Confronto con altri sistemi esistenti

Idealmente il sistema implementato è molto simile ad alcuni in fase di realizzazione (Aura) o ad altri già attivi come quello di VeChain. La similitudine riguarda le funzionalità offerte dai progetti ma non è possibile effettuare un confronto per quanto riguarda l'effettiva implementazione in quanto non sono disponibili alcune informazioni. Nel progetto proposto sarebbero da migliorare l'efficienza degli smart contract e la loro granularità, con l'eventuale aggiunta di altri contratti template, per separare in maniera migliore le operazioni da eseguire. Il sistema ideato non è stato testato in situazioni nelle quali viene effettuata una grande mole di transazioni, quindi non si è dimostrata l'eventuale scalabilità ed efficienza su larga scala.
Un punto di forza del progetto, rispetto ad altri più complessi, è la semplicità e la possibilità di aggiungere in futuro alcune funzionalità per colmare alcune lacune evidenziate in precedenza. Il sistema inoltre funzionerebbe in egual modo nel caso in cui vengano utilizzati RFID tag e nel caso in cui si scelga di usare chip NFC. Gli unici cambiamenti da attuare sarebbero nel modo in cui i clienti dovrebbero leggere le informazioni dal chip ma a livello degli smart contract tutte le funzioni si basano su un codice RFID ed un entità astratta 'Prodotto', di conseguenza non sarebbe necessario modificare il codice. I sistemi più completi sono probabilmente quelli che si basano su una combinazione di blockchain

e architetture tradizionali (dNAS), in questa maniera si può migliorare sia l'efficienza che la sicurezza del progetto, quello proposto si basa solamente sulla blockchain. Il sistema di Aura, al contrario di quello proposto, consente di tracciare il prodotto anche nelle fasi successive all'acquisto da parte dei clienti. Questo è un punto di forza in quanto si riesce a garantire anche durante i processi di manutenzione o riparazione l'originalità del prodotto, aggiornando il certificato di autenticità utile per eventuali rivendite future.

Capitolo 4

Sviluppo applicativo

In questo capitolo verranno descritti gli strumenti utilizzati durante lo sviluppo, gli smart contract che compongono il sistema realizzato e gli script utilizzati per testarlo.

4.1 Strumenti utilizzati per lo sviluppo

Per la realizzazione del progetto si è inizialmente studiato il funzionamento di vari tool e librerie molto diffuse per lo sviluppo di smart contract e per la fase di test. Una volta analizzate le varie opzioni a disposizione è stato scelto come segue l'ambiente di sviluppo:

- Visual Studio Code e Remix IDE: il primo è un IDE utilizzato per scrivere codice in moltissimi linguaggi, con molti plugin ed estensioni e molto comodo da utilizzare; Remix è un IDE open source che è possibile lanciare su un browser oppure come applicazione desktop. Può essere utilizzato per qualsiasi parte dello sviluppo di smart contract, in quanto fornisce la possibilità di compilare, fare il deploy e testare i contratti. Utilizza una blockchain Ethereum-like locale ma è possibile collegarlo ad una blockchain scelta dall'utente e continuare lo sviluppo su quella.

- Ganache: è un emulatore di blockchain Ethereum [8] che permette di generare blockchain locali, settare varie impostazioni e fare il deploy degli smart contract su di esse. Inizialmente per le prove iniziali è stata utilizzata l'applicazione con la GUI per poi passare a linea di comando con contratti più pesanti per evitare rallentamenti.

- Truffle: è un ambiente di sviluppo, un framework di test e un asset pipeline che utilizza la Ethereum Virtual Machine. Alcune funzioni che presenta sono:

  – Compilazione, collegamento, distribuzione e gestione dei binari.

  – Scrittura ed esecuzione di test automatici per i contratti.

  - Framework di implementazione e migrazione dei contratti, anche tramite Node.js.
  - Console interattiva.

- Node.js: Node.js è un runtime JavaScript che permette l'esecuzinoe di codice JS e la creazione di applicazioni server-side. È stato utilizzato principalmente per la scrittura di test e script per interagire con gli smart contract sviluppati sulla blockchain.

- Web3.js: è una raccolta di librerie necessaria per l'interazione con un nodo Ethereum locale o remoto usando HTTP, IPC o WebSocket.

4.2 Struttura del progetto

Il sistema progettato è suddiviso in 4 smart contract e alcune librerie per semplificare certe operazioni. In aggiunta è stato scritto anche un oracolo [9] per poter comunicare ai contratti delle fabbriche i codici RFID da assegnare ai prodotti. Ogni smart contract, escluso l'oracolo, rappresenta un attore nel processo di acquisto e vendita dei capi d'abbigliamento. Le azioni principali svolte durante il processo sono:

- La fabbrica produce un prodotto e lo rende disponibile all'acquisto da parte dei negozi.

- I corrieri autorizzati ritirano i prodotti venduti dalle fabbriche per portarli ai negozi.

- I corrieri possono consegnare il prodotto a loro volta ad altri corrieri autorizzati.

- Il prodotto viene consegnato al negozio e viene messo in vendita per l'utente finale.

- Se durante il processo c'è stata qualche irregolarità, il cliente può chiedere il rimborso che verrà effettuato una volta riconsegnato il pacco al negozio.

Gli attori considerati in questo caso di studio sono le *fabbriche*, i *corrieri* e i *negozi*. Per semplificare la progettazione sono stati considerati solamente i pagamenti eseguiti dagli utenti finali a favore dei negozi e non tutte le transazioni che avvengono normalmente tra corrieri, fabbriche e negozi per l'acquisto e il trasferimento della merce.
L'obiettivo finale del progetto è quello di tracciare i prodotti originali usciti da fabbriche autorizzate per far sì che si possa dimostrare l'autenticità di ciò che si vende e garantire al cliente un certo livello di affidabilità.

4.3 Smart contract utilizzati

4.3.1 Template

Questo smart contract è stato creato per poter usufruire di variabili, funzioni e librerie comuni a tutti i contratti derivati, evitando la scrittura di codice duplicato. Le variabili presenti sono: alcune stringhe usate per il salvataggio di informazioni sulla locazione dell'entità che ha generato lo smart contract; due strutture dati per il salvataggio dei prodotti disponibili nei magazzini delle fabbriche e dei negozi e alcune `struct` utili per poter interagire senza problemi tra i vari contratti. La `struct` più importante è sicuramente `Product`, che contiene tutte le informazioni riguardanti un prodotto. Inizialmente durante lo scambio dei prodotti si inviava direttamente l'istanza di questa struttura, ma per problemi dovuti ad alcune limitazioni di **Solidity** nel passaggio di parametri alle funzioni è stato necessario l'utilizzo di altre 2 `struct` per passare i dati.

```
struct Product{
    string rfid;
    string name;
    string[] rawMaterial;
    address factory;
    mapping(uint => prodTracking) shippers;
    address purchaser;
    uint price;
    uint next_ship_pos;
}
```

Codice 4.1: Product struct

Come si può notare nel codice appena mostrato, la lista di corrieri che hanno posseduto il pacco durante tutto il suo percorso viene memorizzata direttamente nel prodotto. In questa maniera il prodotto è legato indissolubilmente alle informazioni che riguardano chi l'ha posseduto, in quale momento è stato prelevato, chi ha autorizzato il corriere e da chi è stato ritirato.

```
struct prodTracking {
    address shipper;
    address insertedBy;
    address pickedFrom;
    uint timestamp;
}
```

Codice 4.2: prodTracking struct

Esaminando le due librerie presenti in questo smart contract, `SafeMath` è utile per lo svolgimento di operazioni aritmetiche evitando overflow ed altri errori e `arrayStringOp`, serve per evitare la ripetizione di codice identico quando si agisce sugli array di stringhe. Il vantaggio principale derivante dalla presenza di questo contratto è quello di poter modificare alcune funzioni fondamentali come `addProduct` senza dover modificare il codice

anche negli altri contratti, lo stesso vale anche nel caso in cui si debbano aggiungere variabili o campi alle strutture già esistenti. Per questo motivo i tre contratti principali **Factory**, **Shipper** e **Store** ereditano **Template**, anche se nel caso dei corrieri alcune funzioni e variabili non sono utilizzate.
Inoltre sono presenti vari `modifier`, un costrutto di **Solidity** che permette di controllare alcune condizioni specificate, prima dell'esecuzione della funzione chiamata. Questi sono stati utilizzati principalmente per verificare l'identità del chiamante di alcune funzioni, che possono essere chiamate solo dal proprietario del contratto, oppure per segnalare eventuali errori nel caso in cui non vengano inseriti dei parametri necessari alla chiamata della funzione.
Questo smart contract è dichiarato come `abstract` che serve in Solidity per indicare che il contratto contiene funzioni non implementate, perchè personalizzabili dai contratti derivati.

4.3.2 Oracolo

L'oracolo viene utilizzato esclusivamente per distribuire nuovi codici RFID alle fabbriche che li richiedono. È diviso in due parti, la prima è lo smart contract di cui viene fatto il deploy sulla blockchain che viene contattato dai contratti delle fabbriche, la seconda è uno script JavaScript che rimane costantemente in ascolto per intercettare gli eventi lanciati dalla sua controparte on-chain.
Si è deciso di utilizzare un oracolo per far sì che tutte le richieste per i nuovi RFID vengano soddisfatte da un'entità centrale per semplificare la gestione. Gli oracoli sono, per ora, l'unico modo che uno smart contract possiede per richiedere dati non situati sulla blockchain. Ci sono vari step che si susseguono in questo processo di richiesta:

1. Lo smart contract della fabbrica necessita di nuovi RFID quindi effettua una transazione verso un'apposita funzione del contratto dell'oracolo.

2. L'oracolo emette un evento, con le informazioni necessarie per eseguire l'operazione, in modo tale che lo script in ascolto lo catturi.

3. Lo script, una volta catturato l'evento, svolge le operazioni richieste e chiama un'altra funzione dello smart contract dell'oracolo per restituire i risultati ottenuti.

4. Lo smart contract dell'oracolo una volta ottenuti i risultati chiama una funzione di raccolta dati sul contratto della fabbrica che ha effettuato la richiesta e gli trasmette i nuovi RFID.

Esistono sistemi pensati per fornire le API necessarie a fare richieste off-chain dagli smart contract ma per semplicità e per poter personalizzare il codice in questo progetto l'oracolo è stato scritto interamente da me. Nel diagramma sottostante si può osservare uno schema semplificato dell'intero processo.

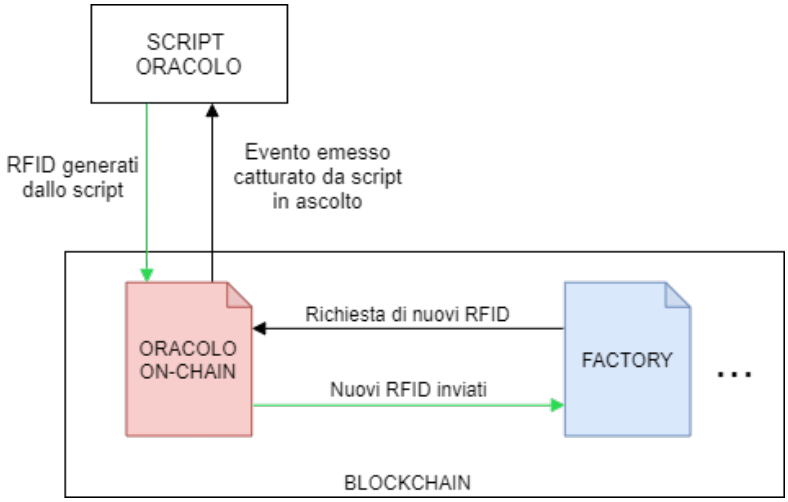

Figura 4.1: Diagramma di utilizzo dell'oracolo

Nel sistema proposto lo script genera come RFID delle stringhe alfanumeriche di 14 caratteri, ma facilmente ampliabili ad un numero maggiore. Difficilmente le stringhe create potranno essere uguali in quanto, per crearle in maniera casuale, è stato utilizzato il modulo `Crypto` di NodeJS che funge da wrapper per le funzioni crittografiche di `OpenSSL`.

```
function generateRfid(quantity) {
  let list = [];
  for(let i=0; i < quantity; i++){
    list.push(crypto.randomBytes(rfidLenght).toString('hex').slice(0,
        rfidLenght));
  }
    return list;
}
```

Codice 4.3: Funzione di generazione degli RFID

Lo smart contract dell'oracolo è molto semplice e presenta solamente due funzioni, la prima per ricevere le richieste dalle fabbriche e la seconda per ritornare i dati una volta che gli RFID sono stati generati.

```
function askForRfids(uint _quantity) external {
    emit rfidEvent(_quantity, msg.sender);
}

function returnRfids(string[] memory _rfidList, address _whoCalled)
   external {
    require(msg.sender == owner, "Not allowed to call this method");
    Factory tmp = Factory(_whoCalled);
    tmp.getNewRfids(_rfidList);
}
```

Codice 4.4: Funzioni dello smart contract dell'oracolo

4.3.3 Factory

La fabbrica rappresenta il luogo dove vengono generati i nuovi prodotti originali di un'azienda e dalla quale sono venduti ai negozi e consegnati ai corrieri. All'interno dello smart contract è presente un riferimento all'oracolo in modo tale che esso possa essere contattato per restituire i codici RFID da assegnare ai prodotti. Automaticamente quando si raggiunge una soglia minima di RFID vengono richiesti nuovi codici all'oracolo, in questo modo si evita il fallimento di transazioni per eventuali errori interni.
Per aggiungere un nuovo prodotto è sufficiente che vengano passate delle informazioni su di esso (il nome e i materiali da cui è composto) alla funzione `addProductFactory`.

```
function addProductFactory(string memory _name, string[] memory _rawMat
    ) external {
     prodTracking[] memory empty; uint len = rfidList.length;
     if(len <= 4){
         oracle.askForRfids(6);
     }
     string memory _rfid = rfidList[len - 1];
     rfidList.pop();
     addProduct(_rfid, _name, _rawMat, address(this), empty, 0);
}
```

Codice 4.5: Funzione per aggiungere prodotti

Nella funzione appena riportata si controlla inizialmente di avere un numero di RFID che non sia inferiore ad una soglia, a quel punto ne viene preso uno dalla lista dei disponibili e viene chiamata la funzione `addProduct` ereditata dallo smart contract **Template**. In questa seconda funzione vengono eseguiti i controlli per verificare il chiamante della funzione e la correttezza dei parametri passati. Nel caso in cui le verifiche vengano superate con successo il prodotto viene inserito con le proprie informazioni nelle strutture dati dello smart contract. Per acquistare i prodotti dalla fabbrica i negozi devono chiamare la funzione `sellProduct` indicando il nome del prodotto desiderato.

```
function sellProduct(string memory _productName) external returns(
    returnProdInfo memory, prodTracking[] memory){
     require(prodCat[_productName].length > 0, "Requested quantity not
         available");
     string memory code = getProdCodes(_productName);
     Product storage result = products[code];
     result.purchaser = msg.sender;
     prodTracking[] memory shippers; returnProdInfo memory values;
     (values, shippers) = getProdInfo(result);
     Product storage tmp = waitShip[code];
     modifyProduct(tmp, values, shippers);
     return getProdInfo(tmp); }
```

Codice 4.6: Funzione per acquistare prodotti

Nel codice 4.6 inizialmente viene controllato che sia disponibile quel prodotto, dopo di che viene prelevato, viene aggiunto nella lista di prodotti in attesa di spedizione e vengono restituiti i dati al negozio.
La maggior parte delle funzioni scritte per interagire con lo smart contract possono essere chiamate solo dal proprietario di esso. Da altri indirizzi è possibile chiamare alcune funzioni per ottenere informazioni riguardanti i prodotti disponibili o per comprare i prodotti dalla fabbrica. Una volta che il prodotto è stato acquistato da un negozio, i corrieri autorizzati dalla fabbrica potranno procedere al ritiro e alla consegna. Si ricorda che per come è stato progettato il sistema, nel momento dell'acquisto da parte del negozio non viene considerato il pagamento della merce.
Per ritirare il pacco, i corrieri chiameranno la funzione `deliverProduct` dello smart contract della fabbrica inserendo il codice del prodotto interessato. In questo momento avviene un controllo da parte del sistema che consente di ritirare il pacco solamente ad un corriere che è stato autorizzato dalla fabbrica in un momento precedente. Dal momento in cui quest'ultima transazione va a buon fine nella fabbrica non sono più presenti informazioni riguardanti il prodotto, che ora è nelle mani del corriere. Il diagramma seguente mostra le relazioni nella quale è coinvolta la fabbrica.

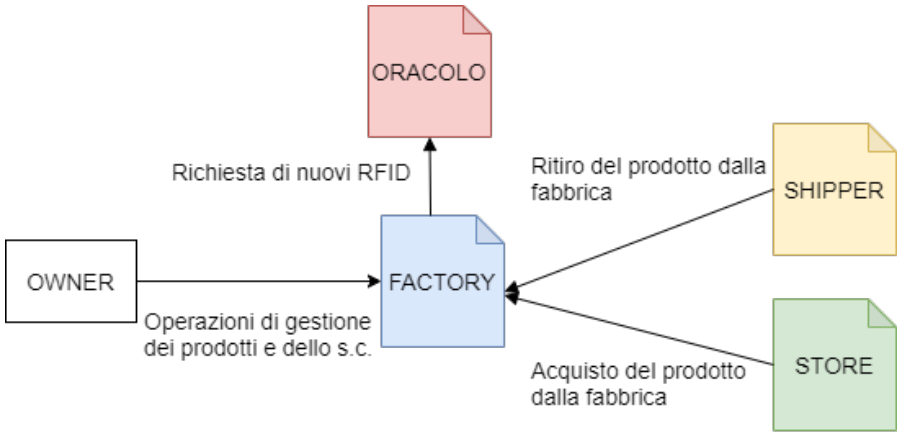

Figura 4.2: Diagramma delle relazioni della fabbrica

4.3.4 Shipper

Questo smart contract è quello dedicato ai corrieri ed è il punto di contatto tra gli altri due smart contract principali (**Factory** e **Store**) in quanto i corrieri devono trasportare la merce dalla fabbrica ai negozi che l'hanno acquistata. Sono previste anche interazioni tra contratti di corrieri diversi per poter consegnare il pacco ad un altro corriere e far effettuare ad esso la consegna al negozio.
Il prodotto entra in possesso del corriere nel momento in cui viene prelevato dalla fabbrica e viene chiamata la funzione apposita dello smart contract che restituisce le informazioni sul prodotto. A questo punto vengono effettuati dei controlli per verificare che la funzione sia chiamata da un corriere autorizzato dalla fabbrica, in caso contrario la transazione viene bloccata. Se le verifiche hanno esito positivo al momento della consegna si salvano nel prodotto i dati del corriere che lo ha ritirato e un timestamp (vedi Codice 4.2) per segnare l'istante in cui è avvenuto lo scambio.
Una volta che il pacco è nelle mani del corriere, come già anticipato, si presentano due possibilità: il corriere consegna direttamente al negozio; il corriere consegna il prodotto ad un altro corriere. Nello smart contract sono presenti due funzioni diverse in base all'azione che si vuole effettuare. Nel primo caso una volta arrivati a destinazione, nel momento in cui viene consegnato il pacco, il negozio deve chiamare una funzione che verificherà che il suo indirizzo corrisponda a quello a cui era destinato il prodotto partito dalla fabbrica e nel caso di risposta positiva restituirà le informazioni sul prodotto. Da questo momento quel prodotto sarà salvato nelle strutture dati del negozio e non più presente nello smart contract del corriere.

```
function deliverPack(string memory _rfid) external returns(
    returnProdInfo memory, prodTracking[] memory){
     require(bytes(onDeliv[_rfid].name).length != 0,"Product not in
         delivery");
     require(msg.sender == onDeliv[_rfid].purchaser, "Not the expected
         recipient");
     **************************
     returnProdInfo memory info;
     prodTracking[] memory shippers;
     (info,shippers) = getProdInfo(onDeliv[_rfid]);
     for(uint i=0; i < onDeliv[_rfid].next_ship_pos; i++){
         delete onDeliv[_rfid].shippers[i];
     }
     delete onDeliv[_rfid];
     return (info,shippers);
}
```

Codice 4.7: Funzione per consegnare al negozio

Nel caso in cui il prodotto debba essere consegnato ad un altro corriere, la seconda funzione controlla che il nuovo corriere sia stato autorizzato a ritirare il pacco e poi restituisce le informazioni del prodotto, dopo aver salvato all'interno del prodotto i dati inerenti a questo scambio.

```
function deliverPackToShipper(string memory _rfid) external returns(
    returnProdInfo memory, prodTracking[] memory){
     bool find; uint ind;
     (find,ind) = findShipper(onDeliv[_rfid], msg.sender);
     require(find,"Not allowed to pick up this product");

     onDeliv[_rfid].shippers[ind].timestamp = block.timestamp;
     onDeliv[_rfid].shippers[ind].pickedFrom = address(this);
                              ...
}
```

Codice 4.8: Funzione per consegnare ad un altro corriere

Le due funzioni variano solamente per i controlli effettuati e per l'aggiunta delle informazioni necessarie al tracciamento nel caso in cui si consegni ad un altro corriere. La parte che si differenzia è riportata nel codice 4.8, il resto rimane invariato. La parte in comune tra le due funzioni serve per ottenere le informazioni del prodotto da consegnare e cancellarle dalle strutture dati dello smart contract del corriere.
In ogni momento si possono ottenere le informazioni su un prodotto che è attualmente posseduto dal corriere inserendo l'RFID in una funzione, che prima di restituire i dati controlla che il richiedente sia la fabbrica da cui è partito il prodotto, il negozio che ha comprato il prodotto o il proprietario dello smart contract.
Nella figura sottostante sono rappresentate in maniera schematica le relazioni dello smart contract **Shipper**.

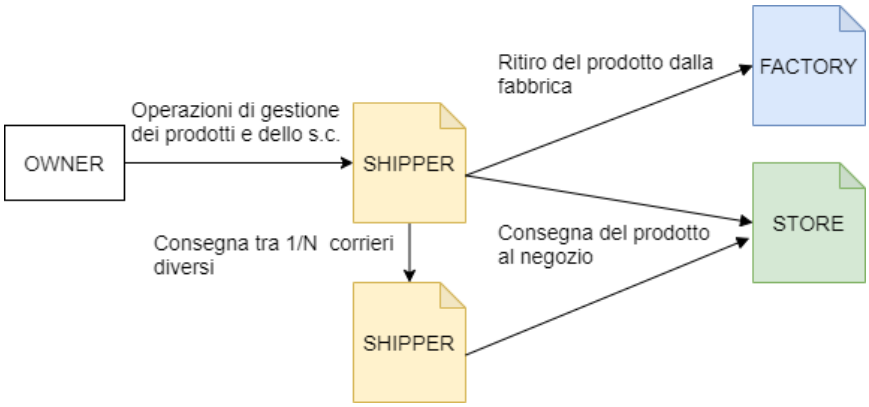

Figura 4.3: Diagramma delle relazioni dei corrieri

4.3.5 Store

Il prossimo smart contract è utilizzato dal negozio per acquistare i prodotti dalla fabbrica, ritirare il pacco dal corriere e vendere i capi d'abbigliamento ai clienti. Il contratto è diviso in due gruppi di funzioni, quelle che servono per l'interazione con i corrieri e le fabbriche e quelle utilizzate per la gestione delle vendite agli acquirenti. Sono state introdotte nuove strutture dati per gestire al meglio le interazioni.

```
struct salesInfo {
    address payable user;
    mapping(uint => Product) purchases;
    uint next_purch_pos;
}
mapping(address => salesInfo) customers;
mapping(address => Product) awaitRefund;
mapping(uint => Product) waitDeliv;
address[] toRefund;
```

Codice 4.9: Variabili e strutture dati aggiunte in Store

Come mostrato nel Codice 4.9 è stata dichiarata una nuova struttura `salesInfo`, questa serve per salvarsi l'indirizzo dei clienti e la lista dei prodotti che hanno acquistato dal negozio. Per la gestione dei rimborsi è stato aggiunto un mapping per salvarsi le richieste di rimborso relative ad un certo cliente e ad un certo prodotto e un array per avere in maniera immediata gli indirizzi di tutti i clienti che hanno effettuato una richiesta.
Il negozio deve inizialmente acquistare i prodotti dalla fabbrica, una volta comprati verranno aggiunti in una lista di attesa fino al momento in cui non saranno effettivamente consegnati dal corriere. Una volta che il prodotto è arrivato al negozio è possibile impostare il prezzo di acquisto in wei e diventa disponibile al pubblico.
Per acquistare un prodotto il cliente deve chiamare la funzione `sellProduct` indicando il nome del prodotto desiderato e inviando i wei necessari per il buon esito della transazione. Ovviamente tutto questo procedimento è semplificato per il cliente in quanto avverrebbe tramite un'interfaccia web o un'app dalla quale ognuno può vedere i prodotti disponibili con le relative informazioni e può acquistarli. In base al prodotto selezionato grazie all'interfaccia verrà poi eseguita la transazione dal sito collegato allo smart contract.

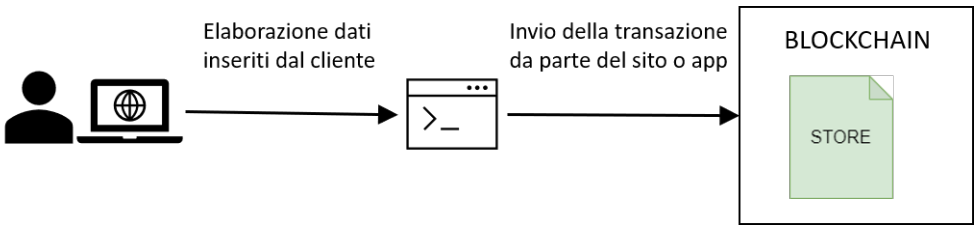

Figura 4.4: Processo di acquisto da parte dei clienti

Al momento dell'esecuzione la funzione controlla che sia stata inviata la corretta quantità di wei per un determinato prodotto e aggiunge il cliente e il relativo acquisto nelle strutture dati illustrate precedentemente.
Come descritto in precedenza, nel caso in cui verificando i dati del prodotto acquistato si scoprono alcune irregolarità sulla merce o trasferimenti tra corrieri non autorizzati, il cliente potrà chiedere il rimborso al negozio. La procedura per il rimborso non è automatica in quanto il negozio dovrà accertare eventuali anomalie prima di restituire i wei all'indirizzo del cliente. Gli step principali sono:

1. Per prima cosa il cliente deve chiamare la funzione `getRefund` passando come parametro l'RFID del prodotto da rimborsare. Se l'utente ha effettivamente acquistato quel prodotto e non ha altre richieste in attesa allora la sua richiesta viene messa in coda in un array.

2. Il proprietario del contratto del negozio a questo punto ha due funzioni disponibili, una che gli mostra tutti gli indirizzi dei clienti che hanno richiesto un rimborso ed un'altra per avere informazioni di un certo prodotto da rimborsare, utile eventualmente per effettuare i controlli.

3. Una volta che i controlli sono stati effettuati e la richiesta di rimborso è stata accettata dal negozio, il cliente dovrà restituire il prodotto e a quel punto viene chiamata la funzione che cancella il prodotto dalla lista degli acquisti del cliente e vengono inviati i wei all'indirizzo del compratore.

Le funzioni che si occupano di ricevere il pagamento e di effettuare il rimborso e le variabili di tipo `address` usate per salvarsi l'indirizzo dei clienti devono essere dichiarate come **payable**, un modifier che consente di inviare denaro ad un certo indirizzo o smart contract durante una transazione. Nello schema sottostante vengono rappresentate in maniera semplici le interazioni dello smart contract appena illustrato.

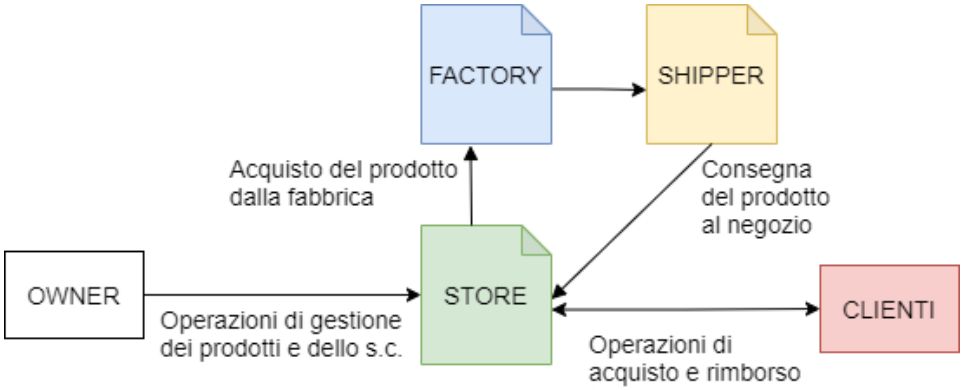

Figura 4.5: Diagramma delle relazioni del negozio

Per concludere viene mostrato uno schema generale con tutte le interazioni tra gli smart contract e le principali funzioni disponibili per le interazioni. Non vengono indicate nello schema tutte le possibili operazioni effettuabili dagli **owner** sui contratti.

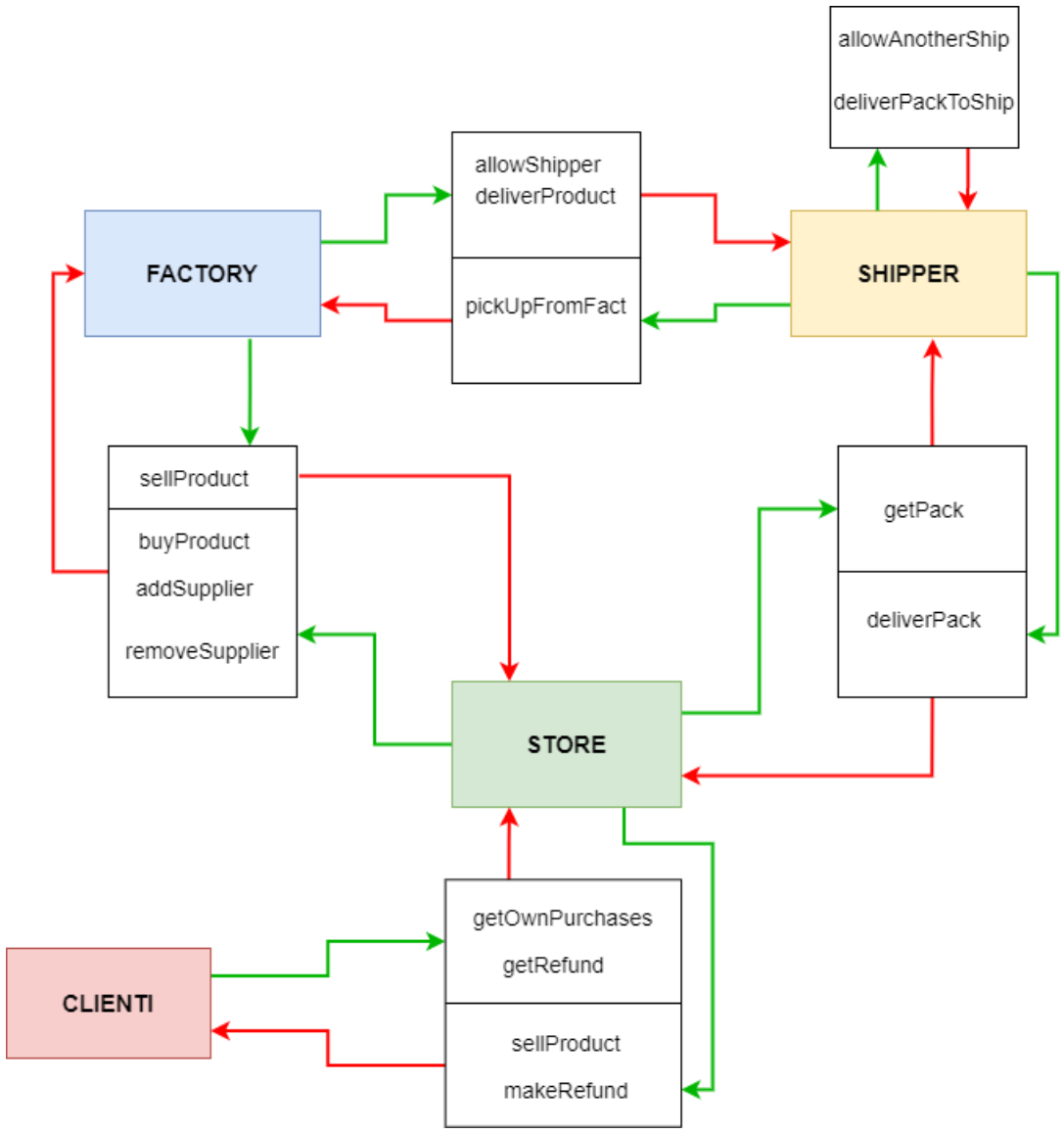

Figura 4.6: Diagramma delle relazioni tra i vari smart contract

Le frecce verdi corrispondono alle funzioni che possono essere chiamate dall'entità da cui partono, le rosse invece corrispondono alle funzioni di uno smart contract che possono essere chiamate dalle altre entità.

4.4 Testing del sistema

La fase di testing è cominciata in contemporanea con lo sviluppo dei primi contratti in quanto il progetto nelle sue fasi iniziali è stato sviluppato su **Remix** che permette un'interazione immediata con gli smart contract.

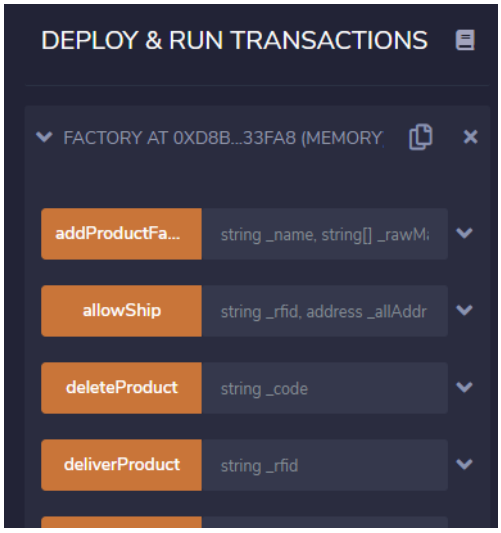

Figura 4.7: Schermata di Remix IDE, area per testare

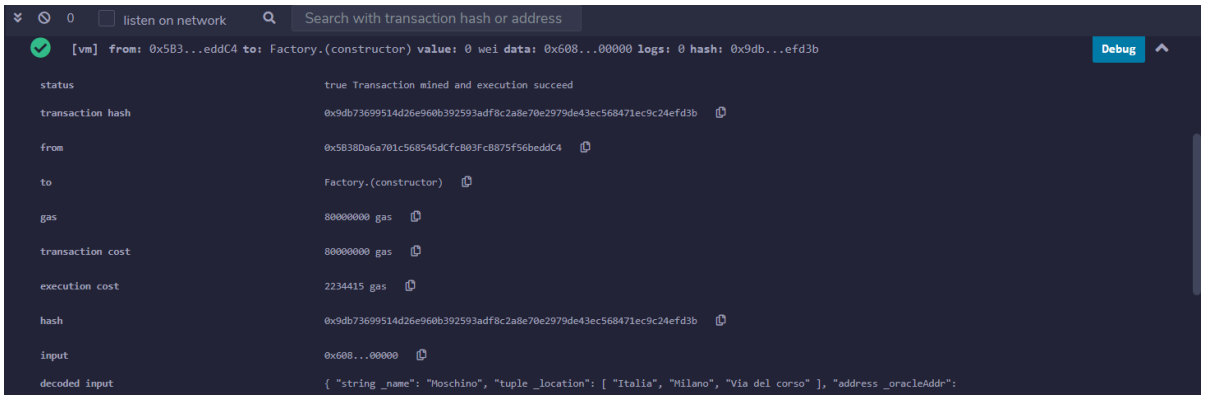

Figura 4.8: Schermata di Remix IDE, area per vedere i dati delle transazioni

Nella zona sinistra della schermata di Remix IDE (figura 4.7) si possono vedere tutti i contratti deployati e si può interagire con essi tramite le funzioni rese disponibili dallo smart contract. Nella parte inferiore dell'interfaccia (figura 4.8) vengono mostrati i dati di tutte le transazioni con eventuali messaggi di errore o dati ritornati dalle funzioni.
Per semplificare il lavoro sono però stati scritti alcuni test automatici in JS eseguibili

con **Truffle** ed alcuni script aggiuntivi per lavorare in maniera più specifica su alcune interazioni tra i vari smart contract. Durante i test non ci si è concentrati sull'efficienza o sull'eventuale risparmio di gas in quanto entrambi non erano tra i requisiti principali del progetto.
Come già anticipato, per simulare una blockchain locale su cui sviluppare e testare gli smart contract è stato utilizzato **Ganache**. La versione da linea di comando è molto leggera ed efficiente quindi è stata preferita all'applicativo più user-friendly per eseguire il progetto completo. Ogni volta che viene lanciato il comando per attivare la blockchain di Ganache viene settato manualmente a 6721000 il *gas limit*, utilizzato per pagare le transazioni, che fallirebbero nel caso in cui esso dovesse essere esaurito prima della conclusione delle operazioni.
Una volta inizializzata la blockchain locale si possono deployare gli smart contract ed interagire con loro tramite la console di Truffle o tramite Remix nel caso in cui ci si colleghi al web server attivato da Ganache. Nello schema sottostante si riassume in maniera molto semplice il procedimento di deploy ed interazione utilizzando gli strumenti e framework illustrati.

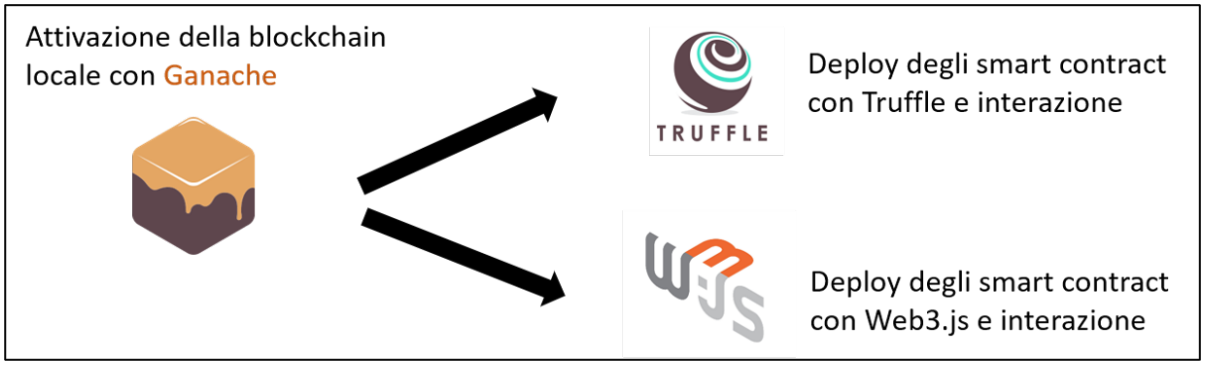

Figura 4.9: Schema semplificato per il workflow

Ora verranno illustrati, divisi per contratti, i vari test effettuati sia in maniera automatica che manualmente.

1. **Test relativi allo smart contract Factory**
   (a) Aggiungere vari prodotti allo smart contract e ottenere le relative informazioni per osservare che essi vengano salvati correttamente nelle strutture dati.
   (b) Verificare che si possano aggiugere nuovi corrieri per il ritiro di un prodotto e che essi possano effettivamente prelevarlo.
   (c) Testare l'invio corretto di nuovi rfid, da parte dell'oracolo, nel caso in cui stiano per terminare.
2. **Test relativi allo smart contract Shipper**

(a) Verificare che si possano aggiungere in maniera corretta nuovi corrieri ai quali consegnare i prodotti.

(b) Effettuare una consegna tra 2 corrieri per mostrare che il prodotto e tutte le informazioni relative vengano passate correttamente.

3. **Test relativi allo smart contract Store**

   (a) Settare il prezzo di un prodotto ed osservare che eventuali transazioni per l'acquisto falliscono se non viene inviata la quantità specificata di wei.

   (b) Generare varie richieste di rimborso da account diversi e vedere come risponde il sistema al salvataggio dei relativi dati.

   (c) Rimborsare i clienti mostrando che sono tornati ad essi i wei depositati per il pagamento del prodotto.

4. **Test relativi alle interazioni tra gli smart contract**

   (a) Controllare per ogni contratto che solamente il proprietario possa chiamare alcune funzioni di gestione e amministrazione dello stesso.

   (b) Comprare tramite uno smart contract `Store` molteplici prodotti da una fabbrica.

   (c) Verificare il processo di consegna tra una fabbrica e un corriere autorizzato.

   (d) Una volta aggiunto un prodotto alla fabbrica, acquistarlo con un negozio, ritirarlo con un corriere autorizzato e consegnare a vari corrieri diversi per mostrare il salvataggio delle informazioni di tracciamento.

Per la creazione di nuovi RFID è stata scritta una nuova funzione all'interno dello smart contract `Factory`, utilizzata solamente durante la fase di testing. Questo passaggio è stato obbligato in quanto lanciando i test automatici con Truffle, non è possibile attivare in contemporanea l'oracolo off-chain necessario per ascoltare gli eventi.
Di conseguenza l'oracolo è stato testato deployando sulla blockchain locale i contratti ed eseguendo uno script JS che automatizzava le chiamate agli smart contract per testare le funzioni.

4.5 Problematiche e punti di forza di Solidity

Solidity è un linguaggio orientato agli oggetti utilizzato per scrivere smart contract su varie blockchain, la più importante è Ethereum. Durante l'implementazione del progetto sono stati riscontrati alcuni problemi e limitazioni dovuti al linguaggio. A differenza dei linguaggi di programmazione abituali Solidity ha alcune limitazioni dovute alla blockchain, per esempio le variabili possono contenere una quantità limitata di dati e i cicli

for sono limitati dal gas, questo meccanismo è stato implementato per evitare codice con cicli infiniti che avrebbe rallentato e danneggiato l'intera rete.
Alcuni punti deboli di Solidity sono la scomodità nello svolgere operazioni su array e stringhe per via della mancanza di una libreria standard e la difficoltà nell'eseguire il debug del codice in quanto gli unici dati che si possono ottenere sono quelli risultanti dalle transazioni. In aggiunta, uno smart contract di cui viene fatto il deploy sulla blockchain non può più essere modificato, quindi è necessaria una fase di test molto rigorosa per evitare problemi di sicurezza. Il linguaggio è nato da poco ed è in via di sviluppo quindi si potrebbero riscontrare alcuni errori ancora da correggere.
Tra i punti di forza del linguaggio invece si ha l'ereditarietà dei contratti, il fatto che sia abbastanza semplice ed intuitivo in quanto influenzato da C++, JavaScript e Python e che sia deterministico. Il linguaggio è deterministico in quanto ogni nodo della rete deve essere in grado di trovare lo stesso risultato dato lo stesso input su un metodo di un contratto. Questo è un vantaggio per gli sviluppatori perchè non devono preoccuparsi del non determinismo in quanto non esistono funzioni non deterministiche nel linguaggio.

Capitolo 5

Conclusioni

La blockchain, come spiegato più volte nel corso della tesi, ha numerosi pregi ma anche qualche difetto, di conseguenza è necessario analizzare la situazione nella quale si vorrebbe utilizzare questa tecnologia per vedere se conviene adottarla oppure no. Utilizzandola in sistemi di tracciamento o anticontraffazione, come quello proposto, si può affermare che siano maggiori i benefici rispetto ad eventuali problematiche riguardanti l'implementazione e i costi iniziali.
Per sua natura la blockchain tiene traccia dei dati in maniera sicura, affidabile e trasparente e di conseguenza sembra essere una soluzione ideale per un processo nel quale la fiducia nei dati e la trasparenza sono fondamentali. Il sistema implementato non è stato testato in situazioni di traffico reale e di conseguenza non se ne conosce l'efficienza ma si è mostrato come potrebbe contrastare alcuni dei metodi di contraffazione esistenti.
Prima di un'eventuale implementazione in una realtà aziendale sarebbero necessarie alcune migliorie ed una fase di test su diverse *testnet* pubbliche di Ethereum.

5.1 Possibili miglioramenti e modifiche

Elenchiamo ora possibili estensioni del progetto per poterlo migliorare ed utilizzare in situazioni reali:

- Estensione del sistema ad altri attori, per esempio l'entità che funge da certificatore dei materiali utilizzati nelle fabbriche e un altro smart contract che si occupa della gestione dei prodotti da ritirare dal mercato.
- Possibilità di inserire, ritirare o comprare più prodotti con una singola operazione per diminuire il numero di transazioni necessarie.
- Gestire i costi delle transazioni effettuate dai negozi per acquistare i prodotti dalle fabbriche e il pagamento dei corrieri per il trasporto della merce.

Bibliografia

[1] C. Tartaglione and F. Gallante, "L'industria del falso e le misure di contrasto alla contraffazione nell'economia moda." https://informatex.it/wp-content/uploads/2017/06/Lindustria-del-falso-e-le-misure-di-contrasto-alla-contraffazione-nelleconomia-Moda.pdf, 2008.

[2] F. Ma, Y. Fu, M. Ren, W. Sun, Z. Liu, Y. Jiang, J. Sun, and J. Sun, "Gasfuzz: Generating high gas consumption inputs to avoid out-of-gas vulnerability," *arXiv*, 10 2019. https://arxiv.org/abs/1910.02945.

[3] M. C. della Volpe, *Le strategie di marketing come strumento di difesa dalla contraffazione nelle imprese del lusso: il caso Pandora*. PhD thesis, Luiss Guido Carli, 2015-2016. https://tesi.luiss.it/17292/1/662201_DELLA%20VOLPE_MARIA%20CAROL.pdf.

[4] N. C. Yiu, "Decentralizing supply chain anti-counterfeiting and traceability systems using blockchain technology," *Future Internet*, vol. 13, p. 84, 03 2021. https://www.mdpi.com/1999-5903/13/4/84.

[5] N. C. K. Yiu, "Toward blockchain-enabled supply chain anti-counterfeiting and traceability," *Future Internet*, vol. 13, no. 4, 2021. https://www.mdpi.com/1999-5903/13/4/86.

[6] "Solution for manufacturers of high-value retail products." https://www.vechain.com/solution/retail, 2019.

[7] A. B. Consortium, "The aura blockchain consortium makes it possible for consumers to access product history and proof of authenticity of luxury goods." https://auraluxuryblockchain.com/?cli_action=1623940580.686#about, 2021.

[8] V. Buterin, "A next-generation smart contract and decentralized application platform." https://ethereum.org/en/whitepaper/, 2013.

[9] V. Mou, "Gli oracoli blockchain spiegati." https://academy.binance.com/it/articles/blockchain-oracles-explained, 2021.

[10] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system." https://bitcoin.org/bitcoin.pdf, 2008.

[11] V. Gramoli, "From blockchain consensus back to byzantine consensus," *Future Generation Computer Systems*, vol. 107, pp. 760–769, 2020. https://www.sciencedirect.com/science/article/pii/S0167739X17320095.

[12] P. Costa, "Implementing a blockchain oracle on ethereum." https://medium.com/@pedrodc/implementing-a-blockchain-oracle-on-ethereum-cedc7e26b49e#, 2019.

[13] M. dello sviluppo economico, "La contraffazione di abbigliamento e accessori: Vademecum per il consumatore." http://www.uibm.gov.it/attachments/disp_abbigliamento.pdf, 2012.

[14] M. Cerlinca, C. Turcu, T. Cerlinca, R. Prodan, and V. Popa, "Anti-counterfeiting iso 15693 rfid solutions involving authentication and traceability using symmetric and asymmetric cryptography," *11th International Conference on development and application system*, p. 4, 2012. http://www.dasconference.ro/cd2012/data/papers/D39.pdf.

[15] J. Ma, S.-Y. Lin, X. Chen, H.-M. Sun, Y.-C. Chen, and H. Wang, "A blockchain-based application system for product anti-counterfeiting," *IEEE Access*, vol. 8, pp. 77642–77652, 2020. https://ieeexplore.ieee.org/document/8985337.

[16] K. Mansour, "Luxury brands using blockchain to fight counterfeiting." https://earlymetrics.com/luxury-brands-using-blockchain-to-fight-counterfeiting/, November 2020.

[17] A. Bhatia, Z. Yusuf, U. Gill, N. Shepherd, M. Kranz, and A. Nannra, "Stamping out counterfeit goods with blockchain and iot." https://www.bcg.com/publications/2019/stamping-out-counterfeit-goods-blockchain-internet-of-things-iot, May 2019.

[18] DGLC-UIBM, "Guida alle tecnologie anti-contraffazione." http://www.uibm.gov.it/attachments/schede%20SOT/Guida%20alle%20tecnologie%20anti-contraffazione.pdf, 2017.

[19] P. Talone, G. Russo, L. Carettoni, and S. Zanero, "Sicurezza e privacy nei sistemi rfid." http://www.rfid.fub.it/edizione_2/Parte_VIII.pdf, 2009.

Ringraziamenti

Ringrazio il mio relatore, il Professor Laneve e il correlatore, la Dottoressa Veschetti per la disponibilità dimostrata e per avermi fatto conoscere una tecnologia innovativa che sicuramente potrà essermi utile in futuro. Un ringraziamento speciale va alla mia famiglia che mi ha sempre supportato in tutto ciò che ho fatto consentendomi di dare il meglio di me e di raggiungere questo importante traguardo. Ringrazio tutti i miei amici di Borgo Tossignano, Imola e Castel San Pietro Terme con cui ho condiviso molte esperienze e riflessioni lungo tutti questi anni. Infine ringrazio tutti i miei amici conosciuti all'università, ho condiviso con loro questa avventura tra risate e momenti di studio e sono sempre stati disponibili ad aiutarmi nel momento del bisogno.