Agenzia nazionale per le nuove tecnologie, l'energia e lo sviluppo economico sostenibile

Ministero della Transizione Ecologica

Ricerca di Sistema elettrico

Analisi e definizione di architetture e componenti per l'uso di dati esterni alla blockchain

F.Bruschi, V.Rana, D.Ghezzi, T.Paulon

POLITECNICO MILANO 1863

SCHOOL OF MANAGEMENT

DIPARTIMENTO DI ELETTRONICA, INFORMAZIONE E BIOINGEGNERIA

Report RdS/PTR(2020)/033

Analisi e definizione di architetture e componenti per l'uso di dati esterni alla blockchain

F.Bruschi, V.Rana, D.Ghezzi, T.Paulon

Aprile 2021

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA
Piano Triennale di Realizzazione 2019-2021 - II annualità
Obiettivo: Tecnologie
Progetto: Tecnologie per la penetrazione efficiente del vettore elettrico negli usi finali
Work package 1: Local Energy District
Linea di attività 1.63: Supporto allo sviluppo di un'applicazione per la valorizzazione del virtuosismo energetico
Responsabile del Progetto: Claudia Meloni, ENEA
Responsabile del Work package: Claudia Meloni, ENEA

Il presente documento descrive le attività di ricerca svolte all'interno dell'Accordo di collaborazione tra l'Agenzia nazionale per le nuove tecnologie, l'energia e lo sviluppo economico sostenibile (di seguito denominata ENEA) e il Dipartimento di Elettronica, Informazione e Bioingegneria del Politecnico di Milano
Responsabile scientifico ENEA: Nicola Gessa
Responsabile scientifico PoliMi: Vincenzo Rana

Indice

SOMMARIO 4

1 INTRODUZIONE 5

2 DESCRIZIONE DELLE ATTIVITÀ SVOLTE E RISULTATI 7

2.1 ORACOLI BLOCKCHAIN: CARATTERISTICHE, TIPOLOGIE, MODELLI DI PROGETTAZIONE E STATO DELL'ARTE 7
2.1.1 Classificazioni degli oracoli Blockchain 8
2.1.2 Modelli di progettazione di oracoli 9
2.1.3 The oracle problem 10
2.1.4 Stato dell'arte 10
2.1.4.1 Feed ad-hoc 11
2.1.4.2 TownCrier 11
2.1.4.3 Oraclize 11
2.1.4.4 Augur 12
2.1.4.5 Kleros 12
2.1.4.6 Chainlink 13
2.1.4.7 Band Protocol 13
2.2 BLOCKCHAIN E BIG DATA: POSSIBILI PROBLEMATICHE, APPROCCI E SOLUZIONI 15
2.2.1 Big Data: un modello di definizione 15
2.2.2 Blockchain e Big Data: possibili approcci e soluzioni esistenti 17
2.2.2.1 Funzioni di hash 18
2.2.2.2 Schemi commit-reveal: un'applicazione evoluta delle funzioni di hash 18
2.2.2.3 Merkle tree 19
2.3 PROOF OF CONCEPT 21
2.3.1 Note tecniche 21
2.3.1.1 Zero Knowledge Proofs 21
2.3.1.2 ZoKrates 22
2.3.1.3 MiMC Hash 22
2.3.2 Proposte iniziali 23
2.3.2.1 Introduzione 23
2.3.2.2 Proposta 1 23
2.3.2.3 Proposta 2 24
2.3.2.4 Proposta 3 25
2.3.3 Evoluzione del progetto 26
2.3.3.1 Scenario definitivo 26
2.3.3.2 Considerazioni per l'implementazione 26
2.3.3.3 Struttura della prova 26
2.3.3.4 Flusso operativo 27
2.3.3.5 Casi d'uso di esempio 28

3 CONCLUSIONI 29

4 RIFERIMENTI BIBLIOGRAFICI 30

4.1 PARAGRAFO 2.1 30
4.2 PARAGRAFO 2.2 31

5 ABBREVIAZIONI ED ACRONIMI 32

6 GRUPPO DI LAVORO 32

Sommario

Il presente documento contiene un riepilogo delle attività svolte all'interno della collaborazione tra ENEA e il DEIB del Politecnico di Milano. Le attività e i risultati sono illustrati nel dettaglio nel capitolo 2: il paragrafo 2.1 è costituito da una panoramica sul tema degli oracoli, comprensiva di un approfondimento sul suo stato dell'arte; il paragrafo 2.2 è dedicato al rapporto tra blockchain e Big Data, e alle possibili soluzioni che si possono adottare per notarizzare on-chain grosse quantità di dati; infine, il paragrafo 2.3 mostra gli step che hanno portato alla definizione dell'ipotesi attuale di Proof of Concept, che prevede l'utilizzo di Zero-Knowledge Proofs per la certificazione del calcolo effettuato da ENEA per determinare il numero di tokens da assegnare agli utenti.

1 Introduzione

Il contributo del Politecnico riguarda la progettazione e lo sviluppo di una Dapp (Decentralized Application) basata sull'utilizzo della tecnologia Blockchain e messa in opera tramite Smart Contract, sviluppata per consentire alla comunità energetica di gestire e valorizzare la flessibilità energetica. Nell'ambito di tali attività, il contributo del Politecnico di Milano riguarda in particolare il collegamento, all'interno della piattaforma oggetto di sviluppo, di una serie di dati esterni, provenienti da diverse fonti, con gli Smart Contract operanti su blockchain, in modo tale che la Dapp sia in grado di garantire e certificare dati e processi, anche in congiunzione con informazioni provenienti dall'esterno della blockchain.

Nell'ambito delle Dapp sviluppate tramite Smart Contract su blockchain, il problema della richiesta, della fornitura e della validazione di dati off-chain, e quindi esterni agli Smart Contract e alla rete blockchain in generale, viene generalmente chiamato "problema degli oracoli".

Esistono diverse tipologie di oracoli che possono essere utilizzati per assolvere tale compito, e la loro classificazione può essere effettuata lungo diversi assi, come ad esempio:

- Oracoli hardware/software

  Gli oracoli hardware forniscono agli Smart Contract dati provenienti dal mondo reale (ad esempio tramite dispositivi IoT), che rilevano direttamente tramite sensori o, più genericamente, dispositivi di lettura delle informazioni. Gli oracoli software trasmettono al network blockchain dati da fonti di informazione online, ad esempio API, siti web o database online

- Oracoli in entrata/in uscita

  Gli oracoli in entrata trasmettono agli Smart Contract informazioni provenienti da fonti esterne, mentre gli oracoli in uscita inviano informazioni contenute nella blockchain, che quindi possono essere lette direttamente dagli Smart Contract, al mondo esterno.

- Oracoli centralizzati/decentralizzati

  Gli oracoli centralizzati costituiscono una fonte unica e quindi, appunto, centralizzata di informazioni per uno Smart Contract. Gli oracoli decentralizzati invece certificano le informazioni comunicate alla blockchain basandosi su più fonti, spesso utilizzando meccanismi di consenso simili a quelli delle blockchain permissionless.

Nell'ambito delle attività di cui al presente documento, emerge la necessità di analizzare e definire l'architettura degli oracoli blockchain più adatta al contesto e ai requisiti funzionali del progetto, di determinare quali informazioni debbano essere trasmesse agli Smart Contract, nonché da quali fonti tali informazioni possano essere recuperate e in che modo possano essere certificate. Tra queste informazioni citiamo ad esempio quelle da utilizzare come benchmark di riferimento per i consumi energetici, necessari agli Smart Contract per determinare gli scostamenti dei consumi degli utenti e le eventuali premialità da assegnare in relazione al grado di virtuosismo energetico raggiunto o raggiungibile.

Le attività in oggetto, ovvero l'analisi delle specifiche e dei vincoli del sistema, il supporto alla progettazione e allo sviluppo del software e l'identificazione delle principali criticità del sistema, sono suddivise e articolate nelle seguenti fasi:

- Fase 1
  - Analisi del sistema
  - Individuazione delle principali criticità, con particolare attenzione al processo di accesso da parte degli Smart Contract ai dati esterni alla rete blockchain

  - Identificazione delle fonti di dati esterni che possono/devono essere certificate
- Fase 2
  - Definizione della comunicazione tra le fonti di dati esterni e gli Smart Contract, con particolare attenzione alla certificazione delle informazioni tramite oracoli blockchain
  - Supporto alla progettazione dell'architettura (hardware/software) di oracoli più adatta al contesto rilevato, con riferimento alle informazioni rilevate durante la fase precedente
- Fase 3
  - Supporto all'implementazione e alla messa in opera di un sistema Proof-of-Concept che dimostri la fattibilità e il funzionamento del sistema di oracoli identificato nelle fasi precedenti

2 Descrizione delle attività svolte e risultati

2.1 Oracoli Blockchain: caratteristiche, tipologie, modelli di progettazione e stato dell'arte

La tecnologia Blockchain si basa sulla decentralizzazione delle informazioni distribuite sui diversi nodi di un network e sulla capacità degli stessi di raggiungere il consenso su uno stato condiviso della rete. Molte piattaforme Blockchain (e.g. Ethereum), oltre allo scambio di asset di valore on-chain, permettono di definire condizioni di esecuzione delle transazioni più complesse e offrono un alto livello di programmabilità, supportando il deployment degli smart contract.

Gli smart contract sono una rappresentazione di clausole contrattuali attraverso un linguaggio software, che vengono eseguite automaticamente in base a condizioni specifiche predeterminate dalle parti. Benché il concetto di smart contract esista già da decenni (teorizzato nel 1997 da Nick Szabo), è il loro sviluppo su piattaforme Blockchain che ne costituisce il caso d'uso più efficace. Infatti, le caratteristiche di questa tecnologia permettono agli smart contract di effettuare computazione in modalità trustless e common knowledge, aggiornando lo stato della rete secondo regole di consenso prestabilite.

Per mantenere il consenso sullo stato della rete, l'esecuzione degli smart contract deve avvenire in modo deterministico e univoco, basandosi unicamente su informazioni in input presenti nel network e condivise tra tutti i nodi. Questo elimina ogni possibile fonte di aleatorietà interna alla rete.

In questo contesto, emerge un limite importante delle Blockchain e degli smart contract: quello di non poter accedere direttamente a fonti di dati esterne al network ("off-chain"), quindi non condivise tra i nodi della rete. Infatti, alcuni servizi e applicazioni Blockchain hanno la necessità di accedere a informazioni provenienti dal mondo reale, spesso integrandoli tra gli input di uno smart contract. Questa esigenza è particolarmente presente nelle applicazioni di business, per le quali è molto sentito il problema di interagire con informazioni rilevanti dal mondo esterno per eseguire uno smart contract.

Si pensi ad esempio a un sistema Blockchain per il tracciamento dei miglioramenti nei consumi energetici di una unità abitativa, in seguito a interventi di efficientamento energetico della stessa. Nella progettazione di uno smart contract che registra e notarizza questi miglioramenti sulla Blockchain, la fonte dei dati di consumo energetico, dalle cui misurazioni dipendono evidentemente le valutazioni sulla efficacia degli interventi di efficientamento effettuati, è un elemento chiave che non deve poter essere sottoposto a manomissione o censura alcuna.

In questo esempio, gli smart contract permetterebbero di garantire soltanto la correttezza di esecuzione del processo di timestamping, non la veridicità dei dati di consumo, che si trovano all'esterno della rete Blockchain. Infatti, lo smart contract deve ricevere i dati come input di una funzione di timestamping contenuta nello smart contract stesso: questo processo introduce un elemento di vulnerabilità rappresentato proprio dalla fonte dei dati, che di per sé non garantisce la validità delle informazioni trasmesse.

Nell'ambito delle Dapp sviluppate tramite smart contract su Blockchain, il problema della richiesta e validazione di dati off-chain viene comunemente affrontato con entità software o hardware chiamati "oracoli".

Gli oracoli Blockchain sono servizi esterni al network che forniscono agli smart contract informazioni off-chain, fungendo quindi da ponte tra le Blockchain e il mondo esterno. Benché la letteratura scientifica non mostri ancora una convergenza su una singola definizione formalmente riconosciuta, la maggior parte dei paper scientifici che trattano l'argomento definiscono gli oracoli come sistemi o entità che permettono la

raccolta, la convalida e la trasmissione di dati da fonti esterne, fornendo alla blockchain informazioni provenienti dal mondo reale.

Gli oracoli forniscono un modo per uno smart contract di ottenere dati e informazioni dal mondo reale, idealmente in modo trustless, ampliandone così notevolmente il campo di applicazione. Inoltre possono essere usati per trasmettere dati in modo sicuro alle Dapp, avvicinando le applicazioni decentralizzate al mondo reale off-chain.

I dati trasmessi dagli oracoli possono essere di varie tipologie. Alcuni esempi comprendono:

1. dati di temperatura o umidità misurati da sensori IoT;
2. informazioni sui consumi energetici;
3. prezzi dell'energia, di asset finanziari o di valute;
4. numeri casuali da fonti di entropia;
5. informazioni sulla ricezione di un pagamento o sullo stato di avanzamento di un processo;
6. dati di geolocalizzazione di un prodotto lungo la filiera produttiva;
7. risultati di eventi sportivi o elezioni politiche.

È opportuno precisare che un oracolo Blockchain non rappresenta esso stesso la fonte di informazioni, ma è piuttosto il livello che verifica e autentica i dati provenienti da fonti esterne e trasmette le informazioni on-chain. Per compiere questo processo, l'oracolo deve essere invocato all'interno di una funzione di uno smart contract, operazione che richiede che siano spese risorse del network.

2.1.1 Classificazioni degli oracoli Blockchain

Esistono diverse tipologie di oracoli Blockchain. Le classificazioni più comuni avvengono principalmente lungo due direzioni: la tipologia di fonte dalla quale i dati provengono e il livello di decentralizzazione dell'oracolo**.**

Ne risulta la seguente classificazione:

- Oracoli *hardware/software/umani*.
  Gli oracoli *hardware* forniscono agli smart contract dati provenienti dal mondo reale, che rilevano direttamente tramite sensori o, più genericamente, dispositivi di lettura delle informazioni. Esempi di oracoli hardware sono sensori IoT o dispositivi facenti parte di impianti domotici, capaci di comunicare autonomamente tramite una connessione Internet i dati misurati.
  Gli oracoli *software* trasmettono al network Blockchain dati provenienti da fonti di informazione online come siti web o database online. Si pensi ad esempio a un sistema API che comunica in tempo reale le tariffe dell'energia elettrica in base alla fascia oraria, per determinare la spesa per l'energia consumata in un dato lasso di tempo.
  Infine, a volte gli individui con conoscenze specializzate in un particolare campo o con responsabilità definite possono servire come oracoli, verificando la loro identità tramite la crittografia e comunicando manualmente i dati alla Blockchain. In questo caso si parla di oracoli *umani*.

- Oracoli *centralizzati/decentralizzati.*
  Gli oracoli *centralizzati* costituiscono una fonte unica e quindi, appunto, centralizzata di informazioni per uno smart contract. Gli esempi di oracoli hardware o sistemi API descritti nel precedente paragrafo sono oracoli centralizzati, poiché costituiscono la singola fonte da cui i dati sono ottenuti. Gli oracoli centralizzati sono più semplici da implementare dal punto di vista

architetturale e permettono a una Blockchain di accedere a dati off-chain in modo agevole. Tuttavia, questi oracoli costituiscono un possibile punto di fallimento del sistema, perché possono essere inaffidabili o malfunzionanti.
Gli oracoli *decentralizzati* invece certificano le informazioni comunicate alla Blockchain basandosi su più fonti, spesso utilizzando meccanismi di consenso simili a quelli delle Blockchain permissionless. Il principale progetto di oracoli di questo tipo è Chainlink, rete decentralizzata composta da molteplici nodi che validano e trasmettono le informazioni richieste dagli smart contract, eliminando la necessità di riporre fiducia in una specifica fonte di dati e superando così i problemi di affidabilità che potrebbero verificarsi utilizzando una fonte centralizzata.

Un singolo oracolo può rientrare in più categorie. Ad esempio, un servizio che ricava individualmente informazioni meteo da un sito web, come la temperatura minima della settimana o l'ora del tramonto di ogni giorno, è un oracolo *software* e *centralizzato*.

2.1.2 Modelli di progettazione di oracoli

Tutte le tipologie di oracoli forniscono alcune funzioni fondamentali. Queste includono la possibilità di raccogliere dati da fonti off-chain, di trasferirli on-chain tramite transazioni firmate crittograficamente e di registrarli in uno smart contract dell'oracolo.

Una volta che i dati sono stati trasmessi on-chain, sono resi disponibili per essere consultati da altri smart contract. Questo avviene attraverso una transazione che innesca una chiamata alla funzione "retrieve" dello smart contract dell'oracolo.

Le principali modalità di configurazione di un oracolo sono le tre seguenti:

- *Immediate-read*: è il design più semplice, che viene utilizzato per ottenere risposta immediata a una domanda semplice. Esempi sono domande chiuse del tipo "la temperatura è maggiore di 20°C?" per le quali la query avviene in modalità just-in-time e solitamente soltanto nel momento in cui la risposta è richiesta.

- *Publish-subscribe*: in cui un oracolo fornisce un servizio di trasmissione per dati che si prevede cambino regolarmente e/o frequentemente. Questa categoria prevede un modello simile a quello di un feed RSS, in cui le informazioni vengono aggiornate continuamente e un segnale viene inviato agli attori considerati "subscribed". Un esempio di dato che risponde a questo modello è un feed di prezzo del gas metano al metro cubo, che può variare giornalmente e deve essere quindi costantemente aggiornato.

- *Request-response*: rappresenta il modello più articolato, in cui il data space è troppo grande per essere immagazzinato nello smart contract dell'oracolo e ci si aspetta che gli utenti siano interessati soltanto a una piccola parte dei dati. Questo modello può essere implementato come un sistema complesso di smart contract on-chain che interagiscono con una infrastruttura dati off-chain, utilizzata per monitorare le richieste e ricevere e restituire i dati.
Il modello request-response è quello comunemente usato nelle architetture client-server, che permette alle applicazioni di avere una conversazione bidirezionale. Per esempio, uno smart contract che dispone i pagamenti agli utenti di una comunità energetica per l'energia inutilizzata e messa a disposizione della rete potrebbe dover richiedere i dati sui prezzi dell'energia su base giornaliera o oraria, secondo uno schema request-response, per garantire che le tariffe applicate siano corrette.

2.1.3 The oracle problem

La necessità per un sistema Blockchain decentralizzato di accedere a dati off-chain apre una possibile falla di sicurezza. Infatti, questo processo introduce la necessità di porre fiducia nei dati che l'oracolo inserisce nella Blockchain ed esiste sempre la possibilità che un oracolo fallisca.

Questo può principalmente accadere in due modi: l'oracolo può essere attaccato o corrotto, trasmettendo quindi dati manipolati che non corrispondono alle reali informazioni fornite dalla fonte oppure, anche se l'oracolo è fidato e non può essere compromesso, esiste la possibilità che i dati su cui sta lavorando siano stati alterati, e che quindi, pur trattandosi di un dispositivo affidabile, esso alimenti gli smart contract con dati non corretti.

La letteratura scientifica definisce il problema degli oracoli in ambito Blockchain come il conflitto di sicurezza, autenticità e fiducia tra oracoli di terze parti e l'esecuzione senza fiducia degli smart contract.
L'origine del costrutto risale a prima che l'ambiente Ethereum fosse lanciato ed è divenuto noto grazie a un celebre post su Reddit. Il blogger Dalovindj, autore del post, si rese conto che durante l'esecuzione di applicazioni sulla Blockchain di Bitcoin che necessitano di informazioni esterne, la verifica dell'affidabilità dei dati estranei alla Blockchain senza alterare il meccanismo di consenso della piattaforma costituiva un problema concreto (*I think of it as "The Oracle Problem"*).

Questo tipo di conflitto si estende ad ogni tipo di smart contract con la necessità di accedere a dati e informazioni non presenti nel network Blockchain sul quale è sviluppato.

Si pensi ad esempio ad una applicazione Blockchain per il tracciamento dei consumi di una comunità energetica e l'addebito dei costi dell'energia consumata. Un'applicazione di questo tipo ha necessità di accedere sia ai dati di consumo degli utenti, che ai prezzi dell'energia delle relative fasce orarie di consumo. Un sistema simile basato su tecnologia Blockchain può offrire al cliente finale la completa trasparenza rispetto ai calcoli sui consumi effettuati dal fornitore di energia che determinano l'addebito in bolletta. Perché questo sia vero però, non basta la trasparenza nel calcolo offerta da uno smart contract, ma è necessario che anche i dati in input siano forniti da fonti affidabili e verificabili.
Rispetto a questo esempio, il problema dell'oracolo assume la forma della domanda: come può un contratto accedere alle informazioni esterne rispetto alla Blockchain su cui è implementato? L'esecuzione degli smart contract deve essere deterministica e controllabile, ovvero deve essere possibile verificare, anche dopo l'accaduto, che un particolare risultato sia effettivamente corretto e aderente al comportamento specificato dal codice del contratto. Questo implica che tutti gli input dello smart contract debbano essere registrati on-chain prima della sua esecuzione.

Esistono diversi approcci per la gestione dei dati esterni in sistemi Blockchain che affrontano il problema dell'oracolo; ogni approccio presenta delle peculiarità che lo rendono più adatto ad alcune applicazioni piuttosto che ad altre.

In seguito vengono descritti i principali approcci attuali, evidenziando per ognuno le principali caratteristiche di funzionamento, i benefici e i rischi associati.

2.1.4 Stato dell'arte

Di seguito sono descritti gli approcci esistenti per interfacciare e collegare l'esecuzione di smart contract con informazioni presenti in ambienti esterni alla blockchain.

2.1.4.1 Feed ad-hoc

I feed ad-hoc sono implementati attraverso smart contract controllati da un'entità che trasferisce informazioni dal mondo esterno attraverso delle transazioni sottomesse alla rete. Ad esempio, un oracolo sul meteo di Milano potrebbe essere uno smart contract che accetta transazioni di aggiornamento solo dal suo proprietario e inoltra le informazioni ai contratti interessati on-chain.

L'oracolo può implementare un modello publish-subscribe, ricevendo sottoscrizioni da che vogliono usare le sue informazioni. L'entità di controllo potrebbe essere una singola persona, un'azienda fidata o un'istituzione pubblica. Ogni volta che l'entità inserirà delle nuove informazioni, l'oracolo le trasmetterà agli smart contract "abbonati" attraverso una transazione oppure, nel caso l'oracolo sia passivo, informerà gli altri contratti del suo stato quando viene interrogato.

2.1.4.2 TownCrier

TownCrier propone un sistema/architettura che permette agli smart contract onchain di richiedere dati web esterni già accessibili tramite TLS e HTTPS. L'architettura di TownCrier è composta da:

- uno smart contract che funge da front-end per gli altri contratti;
- un'enclave: un processo in esecuzione in un ambiente Intel SGX (Software Guard Extensions);
- un relay: un processo che gestisce la comunicazione dell'enclave con la blockchain e con le risorse esterne, fornendo la prova dell'esecuzione sorvegliata di questo processo.

Quando uno smart contract vuole ottenere informazioni da una risorsa web, contatta lo smart contract TownCrier utilizzando le API fornite. Lo smart contract TownCrier codifica la richiesta e informa il processo enclave che, sorvegliato dal relay, genera richieste HTTPS di conseguenza.

Alla ricezione della risposta, l'enclave controlla l'origine dei dati e fornisce un pacchetto di dati firmato in cui certifica la fonte dell'informazione richiesta. Il datagramma viene poi inoltrato dal relay al front-end TownCrier on-chain, e da qui allo smart contract originale richiedente.

I limiti di questo approccio risiedono essenzialmente nel significativo livello di centralizzazione sul gateway TownCrier. Inoltre, anche se è possibile controllare off-chain la corretta esecuzione dell'enclave, la correttezza dei dati forniti non può essere verificata direttamente dallo smart contract originario.

2.1.4.3 Oraclize

Oraclize, ora Provable Things, impiega un'architettura simile a quella di TownCrier, in cui il processo di recupero dati viene attivato da una chiamata allo smart contract, che accede a una fonte di dati esterna e comunica le informazioni al contratto richiedente.

Nel caso di Oraclize, l'autenticità della fonte di dati è provata per mezzo di un TLS-notary, un sistema con cui, dato un revisore fidato, è possibile generare una prova che alcuni dati sono stati effettivamente inviati da una certa fonte durante una sessione TSL. Il revisore è un'istanza di Amazon AWS, che garantisce e firma una prova che i dati sono stati effettivamente generati dalla fonte data e questa prova può essere utilizzata da un attore esterno per verificare la validità dei dati inoltrati da Oraclize a uno smart contract on-chain.

Tuttavia, anche Oraclize ha alcuni svantaggi: ad esempio la prova TSL-notary non può essere controllata direttamente dallo smart contract, a causa della sua complessità. Inoltre, se i dati venissero manomessi il fatto non risulterebbe evidente per lo smart contract, che userebbe i dati come se fossero validi.

Inoltre, Oraclize soffre degli svantaggi di un sistema centralizzato: se viene attaccato, o controllato, può ostacolare il funzionamento degli smart contract che si basano su di esso per l'accesso ai dati esterni.

TownCrier e Oraclize possono essere classificati come oracoli di tipo request-response, poiché permettono agli smart contract di specificare quali risorse vogliono e rispondono con i dati richiesti.

2.1.4.4 Augur

Augur è una piattaforma decentralizzata per la creazione e la risoluzione di mercati di previsione, in cui i partecipanti possono scommettere sul risultato di un determinato evento scambiandosi “shares” sul risultato della predizione (ad esempio, la temperatura a Milano all’alba di domani sarà maggiore di 12°C?). Qualcuno interessato a valutare la probabilità di un evento può creare un mercato di previsione che venda shares di quell'evento, lasciare che gli attori le scambino e osservare il loro prezzo.

Per stabilire automaticamente la verità di un dato risultato, Augur usa un sistema di incentivi: dopo che un evento su cui sta prevedendo è accaduto, un utente può affermare che un certo risultato è quello vero, mettendo in staking un certo valore sotto forma di token di reputazione (REP). Gli altri utenti possono "contestare" l'affermazione iniziale, puntando del valore in token REP su risultati alternativi.

Se il valore in staking da parte degli utenti su un avvenimento differente raggiunge una certa soglia, l’evento viene considerato contestato, il risultato provvisorio viene aggiornato e si apre un'altra possibilità di contestazione per qualche tempo. Questo processo continua fino a quando il risultato provvisorio non viene più contestato e viene quindi considerato corretto dalla maggioranza. Gli utenti che hanno segnalato un risultato scorretto perdono il corrispondente valore in token messo in staking su di esso.

Questi meccanismi incentivano fortemente gli utenti a segnalare della verità, rendendo la piattaforma Augur un oracolo decentralizzato in grado di migrare le informazioni dal mondo reale alla blockchain senza fare affidamento su un intermediario o una terza parte fidata, risolvendo così le problematiche tipiche degli oracoli centralizzati. D'altra parte, il processo di accertamento della verità può essere molto lento, da pochi giorni a diverse settimane, finanche a mesi nel caso di ripetute contestazioni.

2.1.4.5 Kleros

Kleros è un oracolo decentralizzato strutturato come un'organizzazione autonoma decentralizzata (DAO). Il sistema fornisce incentivi economici agli utenti che vogliono agire come giurati, ovvero utenti che forniscono la risposta a domande che vengono poste al fine di guadagnare commissioni in pinakion, token definiti all'interno del sistema. L'idea di Kleros è di costruire un meccanismo in cui i giurati devono coordinarsi per fornire una risposta coerente a una domanda sullo stato del mondo esterno alla blockchain, senza poter colludere segnalando una risposta falsa.

Quando viene posta una domanda, i giurati mettono in staking del valore attraverso i propri token pinakion (PNK). Più alto è il valore in staking, più alta sarà la probabilità di essere selezionati per fornire la risposta. I giurati selezionati votano la risposta che ritengono corretta e generano un commitment sul proprio voto, pubblicando un hash della risposta, del loro indirizzo e di un “salt” segreto (hash(salt+risposta+indirizzo)). Una volta pubblicato il commitment, i giurati non possono più cambiare il loro voto.

Dopo la fase di commitment, i giurati rivelano i loro voti e i token messi in staking vengono ridistribuiti ai giurati che hanno votato l'opzione vincente, cioè quella che ha ricevuto più voti, incentivando così i giurati a votare per la risposta corretta.

Un giurato A potrebbe però tentare di convincere un ristretto gruppo di altri giurati B e C a votare insieme a lui per una risposta falsa. Per comunicare in modo convincente la sua risposta però, dovrebbe rivelare anche

il suo "salt" segreto, per permettere ai giurati B e C di verificare il suo commitment. Tuttavia, se questo accade prima della chiusura della fase di generazione dei commitment, i giurati che vengono a conoscenza del salt segreto possono appropriarsi dei token messi in staking da A per diventare un giurato. In questo modo, eventuali tentativi di collusione tra i giurati vengono disincentivati con la potenziale perdita del valore messo in staking.

Come Augur, Kleros permettere di trovare risposta a domande generali e poste in linguaggio comprensibile all'uomo, ma senza la necessità di creare un mercato di previsioni per ogni evento. Tuttavia, rimangono i limiti dovuti all'alta latenza e al basso throughput del sistema, dovuto al coordinamento degli attori umani.

2.1.4.6 Chainlink

Chainlink è il più diffuso servizio di oracoli blockchain, costituito da una rete decentralizzata di nodi che forniscono dati e informazioni da fonti off-chain a smart contract on-chain.

Quando uno smart contract richiede dati alla rete Chainlink, viene attivato un contratto chiamato Chainlink Service Level Agreement (SLA), che a sua volta genera tre sottocontratti:

- un contratto *Chainlink Reputation*, che controlla il track record dei nodi Chainlink fornitori di dati per verificarne l'affidabilità;
- un contratto *Chainlink Order-Matching*, che inoltra la richiesta di dati ai nodi oracolo ritenuti affidabili e ne seleziona alcuni per soddisfare la richiesta;
- un contratto *Chainlink Aggregating*, che riceve tutti i dati forniti dai nodi oracolo scelti e li convalida e/o li riconcilia per ottenere un risultato accurato.

I nodi oracolo della rete Chainlink che ricevono la richiesta di dati off-chain del contratto richiedente, usano il software Chainlink Core per tradurre la richiesta dal linguaggio di programmazione on-chain (ad esempio Solidity sulla blockchain Ethereum) a un linguaggio di programmazione che una fonte di dati del mondo reale può interpretare.

Questo passaggio permette di indirizzare la richiesta ad un'interfaccia esterna (ad esempio delle API web) che fornisce i dati richiesti. Una volta che i dati sono stati raccolti, vengono nuovamente tradotti nel linguaggio on-chain attraverso Chainlink Core e infine inviati di nuovo al contratto di aggregazione Chainlink.

Gli utenti che richiedono le informazioni off-chain utilizzano il token LINK per remunerare gli operatori dei nodi Chainlink per il loro lavoro. I prezzi vengono fissati dall'operatore del nodo Chainlink in base alla domanda dei dati che possono fornire e al mercato attuale per quei dati.

I token LINK vengono poi messi in staking dagli operatori dei nodi Chainlink e agiscono da incentivo per il corretto funzionamento della rete. Infatti, i nodi con una quota maggiore di LINK in staking hanno maggiore probabilità di essere scelti per soddisfare le richieste (e quindi guadagnare altri token LINK), mentre i nodi disonesti vengono puniti con la perdita di parte della loro quota di LINK se offrono dati sbagliati o di qualità scadente.

2.1.4.7 Band Protocol

Band Protocol è una piattaforma di oracoli decentralizzata, costruita su una propria blockchain BandChain, che consente di aggregare e collegare dati ed API del mondo reale a smart contract, fungendo quindi da oracolo per questi ultimi.

Ogni utente possessore di token BAND può diventare un fornitore di dati. Per fare ciò, l'utente mette in staking attraverso uno smart contract sulla BandChain i token che soddisfano il requisito minimo del

contratto e che servono come garanzia. In questo modo l'utente si propone come candidato fornitore e può inserire i dati off-chain all'interno del protocollo.

Per accedere ai dati esterni, le Dapp interagiscono direttamente con gli smart contract del protocollo Band, pagando una fee per accedere alle fonti di dati gestiti dalla comunità. Ogni volta che i dati vengono richiesti, le fee pagate vengono distribuite ai relativi fornitori.

Per incentivare gli utenti a fornire dati corretti, il protocollo prevede un sistema di incentivi. Attraverso lo staking dei token BAND, gli utenti della comunità possono votare i fornitori di dati migliori attraverso un processo di delega simile a quello impiegato nei meccanismi di consenso. Gli utenti che sostengono i fornitori di dati più affidabili ricevono una quota delle fee pagate dagli smart contract esterni per accedere alle informazioni, mentre i nodi disonesti vengono puniti con la perdita parziale o totale dei loro token BAND se forniscono dati scorretti alla rete.

Questo meccanismo assicura che tutti gli attori coinvolti nella creazione di un set di dati abbiano un incentivo economico coordinato per fornire dati di qualità.

2.2 Blockchain e Big Data: possibili problematiche, approcci e soluzioni

In alcuni contesti le applicazioni Blockchain-based sono chiamate a gestire una grande quantità di dati, provenienti da diverse fonti. Nei casi d'uso che prevedono l'interazione degli smart contract e degli oracoli Blockchain con quantità di dati ingenti e/o non strutturati è necessario avere qualche accortezza in più, in quanto la gestione dei dati *offchain* può diventare più complessa.
Di seguito verranno illustrati alcuni approcci comunemente adottati in applicazioni Blockchain, che permettono di superare alcune criticità che possono manifestarsi nella gestione di dati esterni alla Blockchain in contesti di Big Data.

2.2.1 Big Data: un modello di definizione

È innanzitutto necessario chiarire il quadro definitorio, ovvero cosa si intende con Big Data e come distinguere fonti di dati di questo tipo da sorgenti più tradizionali di informazioni.

Non esiste una soglia di riferimento prestabilita in termini di dimensione oltre la quale è lecito parlare di Big Data: in genere il termine Big Data si riferisce ad un dataset la cui dimensione va al di là delle capacità di un database tradizionale di memorizzare e gestire i dati. Si parla di big data quando la dimensione e la complessità dell'insieme di dati è tale da richiedere la definizione di nuovi strumenti e metodologie per estrapolare, gestire e processare informazioni.

L'espressione Big data viene spesso associata all'aumento delle fonti informative e più in generale dall'evoluzione di tecnologie come ad esempio l'Internet of Things (IoT), che abilita la raccolta e l'utilizzo di una grande quantità di dati da una moltitudine di sorgenti diverse.

È importante sottolineare come la definizione di Big Data non si limiti solamente a descrivere grandi volumi di dati, ma indica anche l'aumento della complessità e della varietà delle informazioni raccolte in un dataset. Questi aspetti vengono affrontati dalla letteratura scientifica facendo generalmente riferimento a un modello divenuto celebre per la semplicità ed efficacia nella definizione delle dimensioni di analisi chiave che concorrono a definire i Big Data.

Si tratta del Modello delle 3V dei Big Data, descritto per la prima volta dall'analista di Gartner Doug Laney nel 2001, che introduce tre vettori di analisi: Volume, Velocità e Varietà.

Volume

Il volume è la prima dimensione di analisi che più intuitivamente viene associata ai Big Data. In effetti, parlando di Big Data è facilmente intuibile che si faccia riferimento a sorgenti di informazioni che producono grandi volumi di dati.
Quando si parla di volume in contesti di Big Data però, ci si riferisce a quantità di dati incredibilmente grandi. Con l'avanzamento della tecnologia, avremo la possibilità di raccogliere quantità sempre maggiori di informazioni: per esempio, l'utilizzo di sensori connessi e intelligenti permette di effettuare misurazioni sempre più frequentemente. Ad esempio, rilevando la temperatura di un ambiente una volta al minuto si otterranno 525.600 misurazioni in un anno, da un solo sensore. Supponendo di avere una fabbrica con mille sensori, si tratterebbe di oltre mezzo miliardo di punti dati che compongono le curve di temperatura degli ambienti monitorati.
Questo esempio si riferisce alla misurazione di una sola grandezza, la temperatura. Come è facilmente intuibile, se fosse necessario misurare contemporaneamente altri parametri, la grandezza del dataset aumenterebbe drasticamente.

Dunque, è chiaro che il volume dei dati è un fattore rilevante ed è infatti il primo vettore di definizione dei Big Data.

Velocity

Per velocità si intende invece la rapidità con cui i dati vengono acquisiti e utilizzati. I dati fluiscono da fonti come macchine, contatori o sensori che sono in grado di aumentare la granularità di misurazione fino a più volte al secondo, creando un flusso costante e continuo di informazioni.
Prendendo in esame una piattaforma di social network, ad esempio, dove milioni di utenti che caricano giornalmente (e a tutte le ore) contenuti di vario tipo come foto, video e testi, la velocity è definita dalla velocità stessa con la quale questi dati vengono trasferiti dai dispositivi degli utenti ai server dell'azienda.
La velocity è proprio la misura di quanto velocemente vengono ricevuti i dati che devono essere gestiti. I server di un social network non devono solamente ricevere i contenuti degli utenti, ma devono essere in grado di elaborarli, archiviarli ed essere in grado di recuperarli in seguito.
La velocità di trasferimento dei dati è quindi il secondo vettore di analisi rilevante per la definizione di Big Data.

Variety

La dimensione di variety è legata alle molteplici tipologie di dati disponibili che, a causa della diversità delle fonti di generazione dei dati, si riferiscono sia a dati strutturati che non strutturati.
I dati strutturati sono dati creati utilizzando uno schema predefinito e sono tipicamente organizzati in un formato tabellare; al contrario, i dati non strutturati hanno spesso formati differenti e non vengono generati secondo una struttura predefinita, per questo non sono adatti ad essere conservati in un database relazionale.
I Big Data fanno riferimento a dati generalmente non strutturati ed estremamente vari. Per questo la varietà dei dati è una terza dimensione significativa per la definizione di Big Data.

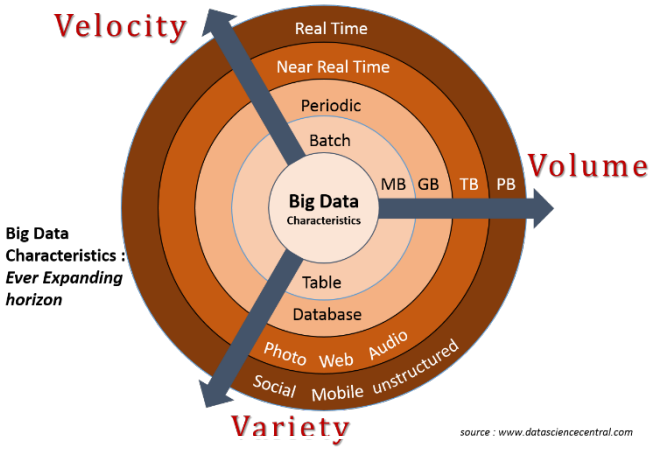

Figure 1: Le 3V dei Big Data.
Fonte: https://www.datasciencecentral.com/

Benché ad oggi il Modello delle 3V sia ampiamente accettato per la definizione di Big Data, il paradigma di Laney è stato arricchito dall'aggiunta di due ulteriori variabili che esprimono le dimensioni di Veridicità (Veracity) e Valore (Value).
Per questo, con l'evoluzione del modello, si parla di 5V dei Big Data.

Veracity

La Veridicità si riferisce al livello di qualità, integrità e accuratezza dei dati, che devono essere affidabili e raccontare il vero. L'avvento dei Big Data ha aumentato drasticamente la velocità con la quale i dati vengono raccolti e sono aumentate le fonti. La qualità e l'integrità delle informazioni rimane però un pilastro imprescindibile per ottenere analisi che siano utili e affidabili.
A sottolineare l'importanza di questo concetto, nel settore è nota l'espressione: "Bad data is worse than no data".

Value

Negli ultimi anni i Big Data sono stati definiti come "il nuovo petrolio", un'espressione divenuta ormai celebre che indica come i dati possano rappresentare una fonte inestimabile di valore per aziende e organizzazioni. Tuttavia, limitarsi a raccogliere i dati non garantisce di per sé la possibilità di estrarne conoscenza, che è invece il risultato di un processo di analisi dei dati raccolti. Per attuare questo processo, e far sì che dai Big Data possano essere ricavate informazioni utili, sono necessari strumenti di Big Data Analytics.
Il valore si riferisce quindi a quanto sono utili i dati nel processo decisionale e sottolinea l'importanza di estrarre conoscenza dei Big Data usando analisi adeguate.

Figure 2: Le 5V dei Big Data.
Fonte: https://suryagutta.medium.com/the-5-vs-of-big-data-2758bfcc51d

2.2.2 Blockchain e Big Data: possibili approcci e soluzioni esistenti

In alcuni contesti le applicazioni Blockchain devono dialogare con fonti di dati esterne che producono grandi volumi di dati. Questo aspetto pone delle sfide nella progettazione di oracoli che dai Big Data devono estrarre e comunicare alcune informazioni puntuali e rilevanti per gli smart contract.
Il problema si pone anche in relazione alla scarsità e al costo di archiviazione dei dati on-chain. Lo spazio a disposizione sulla Blockchain è limitato e ha un elevato costo computazionale, perciò è in generale sconveniente caricare direttamente sulla Blockchain i dati che sono di interesse.

Di seguito vengono quindi illustrati alcuni approcci tecnici adottati da applicazioni Blockchain in contesti di Big Data, che permettono una agevole ed efficace verifica dei dati notarizzati su Blockchain.

2.2.2.1 Funzioni di hash

L'approccio più comune per ovviare alla scarsità di spazio disponibile on-chain prevede l'applicazione di una funzione di hash che produca una stringa alfanumerica di lunghezza determinata del dato in input, chiamata *hash code*, *digest* o semplicemente *hash* del dato, costituendo l'impronta digitale del dato stesso.
Caricare l'hash del dato sulla Blockchain permette di fatto una notarizzazione di quel dato, senza però caricarlo integralmente sulla Blockchain - cosa che richiederebbe risorse rilevanti e, nel caso di Big Data, sarebbe impossibile.
Le funzioni di hash trovano ampio utilizzo nelle tecnologie Blockchain e Distributed Ledger, sia per la notarizzazione di dati e documenti, come detto, ma anche nei meccanismi di consenso. Un esempio è l'algoritmo Proof of Work - il meccanismo di consenso attualmente utilizzato da Bitcoin ed Ethereum - che basa il processo di validazione dei blocchi di transazioni proprio sulle funzioni di hash.
Tuttavia, se parliamo di Big data, l'approccio di notarizzazione sopra descritto, benché permetta di certificare l'esistenza anche di grandi quantità di dati con un singolo hash, presenta dei limiti importanti in eventuali processi successivi di audit o verifica di informazioni puntuali contenute nei dati notarizzati.
In altre parole, la notarizzazione di un hash generato a partire da diversi gigabyte di dati permette, in una verifica successiva, di essere sicuri che tutti i dati in input all'hash notarizzato su Blockchain non siano stati modificati o manomessi.
A titolo di esempio, si potrebbe notarizzare l'hash dei consumi orari giornalieri degli utenti per poi premiare chi non abbia superato una certa soglia totale nei weekend; ogni utente sarebbe in grado di provare, a partire dai consumi del proprio contatore, di non avere modificato i consumi a posteriori e di avere diritto a ricevere il premio.

2.2.2.2 Schemi commit-reveal: un'applicazione evoluta delle funzioni di hash

La caratteristica fondamentale di una Blockchain public-permissionless è che il suo stato, le transazioni e l'esecuzione degli smart contract avviene in totale trasparenza per tutti gli attori, anche esterni al network.
La trasparenza può però essere in antitesi con le necessità di privacy riguardo i dati on-chain, soprattutto per applicazioni in ambito aziendale. Poiché tutte le transazioni in una Blockchain permissionless sono pubbliche, vengono usati dei meccanismi per tenere alcune informazioni temporaneamente nascoste.
Uno di questi meccanismi è rappresentato da uno schema commit-reveal o commitment scheme, ovvero un algoritmo crittografico che sfrutta le funzioni di hash per permettere a qualcuno di "impegnarsi" in un valore, mantenendolo segreto per agli altri utenti, con la possibilità di rivelarlo in seguito.
Oltre ai vantaggi in termini di privacy, l'applicazione di schemi di commitment di questo tipo permette di velocizzare anche nelle fasi di verifica successiva e puntuale dei dati.
Lo schema ha due fasi: una fase di impegno (o commitment) di un valore, un documento o un insieme di dati, e una fase di rivelazione (o reveal) in cui il valore viene rivelato e controllato. I valori impegnati in uno schema di commitment sono vincolanti, il che significa che non possono essere modificati, ritrattati o sostituiti una volta impegnati.
Questo approccio può essere utilizzato in una applicazione Blockchain-based che registra su Blockchain i consumi energetici degli utenti per determinare delle premialità legate a comportamenti energetici virtuosi. In questo scenario, i dati sui consumi rappresentano informazioni sensibili dei clienti, pertanto non si vuole che gli utenti possano venire a conoscenza dei consumi dei clienti semplicemente dall'osservazione delle transazioni sulla Blockchain. Allo stesso tempo, però, è necessario che sia prevista la possibilità per alcuni attori di accedere in chiaro alle informazioni notarizzate su Blockchain e verificare i dati sui consumi (ad

esempio le Forze dell'Ordine che debbano effettuare controlli, oppure gli utenti stessi che vogliano verificare che nel sistema siano registrati i dati corretti che li riguardano).
L'applicazione di uno schema commit-reveal prevede che venga registrato sulla Blockchain il solo hash dei dati di consumo dell'utente, creandone così una prova di esistenza (*Proof-of-Existence*) in un determinato istante temporale, che coincide con l'istante di validazione del blocco che lo contiene (*timestamping*). La prova di esistenza stessa funge da commitment per i dati da parte dell'utente.
In questo modo, i dati sensibili non hanno bisogno di essere caricati né condivisi, l'unica informazione che viene divulgata è il loro hash.
Ciò permette, da una parte, di superare le difficoltà di gestione di grandi moli di dati sulla Blockchain; dall'altra, consente di mantenere la riservatezza sui dati, garantendo allo stesso tempo la loro esistenza e autenticità. Se questi venissero manomessi o modificati infatti, il relativo hash risulterebbe completamente diverso e differente da quello registrato sulla Blockchain.
In un momento successivo, quando dovesse sorgere la necessità per un attore di verificare che i dati forniti da un utente siano effettivamente quelli corretti, l'utente invierà i dati richiesti e, applicando nuovamente la stessa funzione di hash impiegata nella fase di commitment, si potrà provare che quelli inviati sono effettivamente i dati su cui si è fatto commitment sulla Blockchain, poiché l'hash risulterà uguale al valore impegnato.
Uno schema commit-reveal permette quindi di preservare la privacy dei dati on-chain, garantendo allo stesso tempo la possibilità di una verifica successiva. Tuttavia, questo sistema rende comunque ancora molto difficile verificare una singola informazione all'interno dei dati notarizzati, in quanto questa verifica dovrebbe essere fatta manualmente.
Per migliorare il processo di verifica puntuale di informazioni contenute in grandi moli di dati su cui si è fatto commitment, è possibile utilizzare una particolare struttura dati creata con l'obiettivo di facilitare la verifica di grandi quantità di informazioni, come ad esempio i merkle tree.

2.2.2.3 Merkle tree

I merkle tree rappresentano la struttura dati in cui vengono organizzate le transazioni quando vengono inserite all'interno di un blocco per essere facilmente riconoscibili, distinguibili, rintracciabili e verificabili a posteriori.
Come suggerisce il nome stesso, i dati in un merkle tree sono organizzati in una struttura ad albero. Nello specifico, le informazioni vengono raggruppate a coppie in una struttura piramidale, calcolando l'hash di ogni coppia e proseguendo progressivamente a livelli più alti dell'albero, fino a risalire l'intera struttura piramidale, calcolando infine l'hash complessivo di tutto l'albero. Quest'ultimo hash viene chiamato radice (*hash root*).

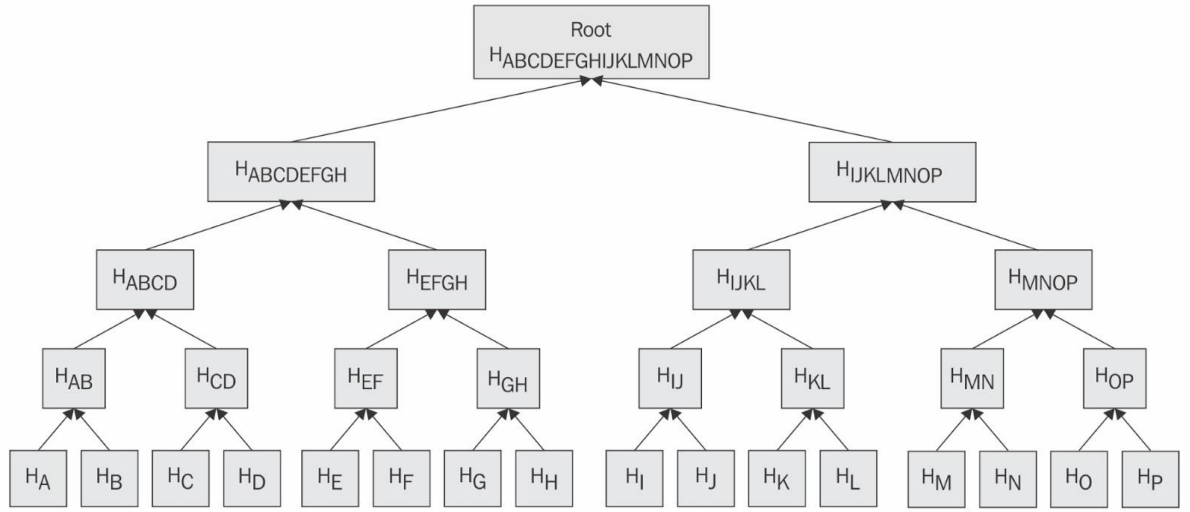

Figure 3: La struttura di un Merkle Tree.
Fonte: https://www.oreilly.com/

In ambito Blockchain, i merkle tree hanno riscosso particolare successo grazie ad alcune loro peculiarità. Per calcolare il root a partire da una foglia sono sufficienti solamente gli hash dei sottoalberi necessari a risalire l'albero dal basso verso l'alto, che sono in numero minore rispetto al numero totale di foglie; questo permette di verificare velocemente se una foglia è contenuta all'interno di un root (la cosiddetta "*merkle proof*"), senza la necessità di dover ricalcolare tutto l'albero.
Questa struttura dati inoltre è particolarmente adatta a certificare grossi volumi di informazioni come per esempio dati biometrici, consumi energetici, rilevazioni di sensori, ecc. perché preserva la verificabilità e allo stesso tempo richiede un consumo limitato di risorse computazionali e di storage.
Molti Blockchain networks esistenti utilizzano questa struttura per registrare le transazioni all'interno dei blocchi. In Bitcoin, ad esempio, le transazioni di un blocco costituiscono le foglie di un merkle tree binario che utilizza la funzione di hash SHA-256; il root dell'albero viene salvato nell'header del blocco (unica parte custodita dai client lightweight), mentre l'intero albero viene conservato nel body. Per i motivi sopracitati, la scelta di questa struttura dati rende veloce verificare se una transazione è contenuta in un determinato blocco, mantenendo limitate le dimensioni.
Questo metodo permette quindi di registrare su Blockchain la prova di esistenza di grandi volumi di dati, consentendo allo stesso tempo una agile verifica di informazioni puntuali su di essi.

2.3 Proof of Concept

2.3.1 Note tecniche

2.3.1.1 Zero Knowledge Proofs

Un ulteriore strumento crittografico che è possibile utilizzare in ambito Blockchain e Big Data sono le dimostrazioni a conoscenza zero, o zero knowledge proof (ZKPs).

Le zero knowledge proof interattive sono dei protocolli che consentono ad un attore di dimostrare la veridicità di un'affermazione, senza dover rivelare altre informazioni oltre alla veridicità della stessa.
Questa tipologia di prove è uno strumento largamente usato in crittografia. Tuttavia, l'interattività richiesta da questi protocolli non sempre è possibile tra gli attori che devono produrre e verificare le prove. Questo aspetto rappresenta un limite, in particolare all'interno di un processo basato sulla tecnologia Blockchain.

Proprio per superare questa problematica, partendo dalle prove zero knowledge interattive sono stati sviluppati protocolli non interattivi, che consentono di produrre la prova sotto forma di un certificato che può essere utilizzato da chiunque, anche senza la collaborazione del suo creatore, per convincersi della fondatezza della dichiarazione provata. I certificati creati in questo modo sono efficaci nel dimostrare la veridicità di un'affermazione, ma a discapito dell'efficienza e semplicità di calcolo. Si tratta infatti di prove matematiche lunghe e complesse sia da calcolare che da verificare.

Con l'obiettivo di rendere questi protocolli utilizzabili in ambiti pratici, una ulteriore evoluzione di questi strumenti ha portato la definizione dei protocolli zk-SNARK.
L'acronimo zk-SNARK corrisponde a "Zero-Knowledge Succinct Non-Interactive Argument of Knowledge" e si riferisce a una costruzione di prove in cui si può dimostrare il possesso di un'informazione dichiarata, senza rivelare tali informazioni e senza la necessità di alcuna interazione tra il dichiarante e il verificatore. Si tratta infatti di un singolo messaggio inviato dal generatore della prova al verificatore, che può essere validato molto velocemente e senza necessità di ingente potenza computazionale.

Questo genere di prove è molto utilizzato in applicazioni privacy-preserving, perché permettono di mantenere private alcune informazioni ritenute sensibili. Un esempio classico è quello del negozio di liquori online: al posto di imporre agli utenti di caricare un documento d'identità che contiene dati sensibili non necessari al fine di dimostrare di essere maggiorenni, è possibile utilizzare una ZKP che dimostra che l'utente ha più di 18 anni, senza dare alcuna informazione sulla data di nascita. Un altro esempio per cui vengono utilizzate di frequente è dimostrare la conoscenza di una foglia contenuta in un merkle root, senza fornire alcuna informazione sulla foglia stessa.

Se per esempio i consumi giornalieri di un mese vengono organizzati in un merkle tree, gli utenti possono utilizzare una ZKP per dimostrare alcune caratteristiche dei loro consumi in un determinato giorno (es. inferiori a una certa soglia) senza rendere pubblici i consumi stessi.
Nell'ambito Blockchain, questi protocolli trovano ampie possibilità di applicazione. Ad esempio, possono dimostrare crittograficamente la correttezza e validità di una transazione, senza rivelare alcuna informazione come gli indirizzi del mittente, del ricevente o l'ammontare transato. Oppure possono essere impiegati per generare una prova, verificabile da uno smart contract, che un utente è in possesso della pre-image di un determinato hash o della chiave segreta che ha firmato una specifica transazione, senza rivelarla.

2.3.1.2 ZoKrates

ZoKrates è uno strumento che rende più agevole l'utilizzo di ZKP in applicazioni blockchain, e più in particolare nell'ecosistema Ethereum. Questo toolbox offre un linguaggio di programmazione ad alto livello (non Turing completo), simile per certi versi a Python, che permette ai developers di generare zkSNARKs con una tecnica che non si discosta molto dalla programmazione a cui sono abituati.

Per generare una prova con ZoKrates per prima cosa è necessario scrivere un programma con questo linguaggio, in cui vengono descritte le caratteristiche della prova. In questa fase viene anche specificato quali input del programma devono essere pubblici e quali privati, in modo da stabilire quali informazioni saranno derivabili dalla prova e quali no. Zokrates inoltre mette a disposizione degli sviluppatori una libreria standard, che contiene alcune funzioni di hash e alcune funzioni di utilità generale. Una volta scritto il programma è possibile utilizzare i comandi offerti da ZoKrates CLI:

- **ZoKrates compile**: compila il programma, traducendolo nel circuito matematico da esso descritto
- **ZoKrates compute-witness**: a partire da un circuito ZoKrates e da una serie di input produce un cosiddetto witness, ossia una allocazione di valori di ingresso valida al fine della generazione della prova
- **ZoKrates setup**: a partire da un circuito ZoKrates genera proving key e verification key
- **ZoKrates generate-proof**: a partire da un circuito ZoKrates, da un witness e da una proving key genera una prova in formato JSON
- **ZoKrates export-verifier**: a partire da una verification key crea uno smart contract scritto in Solidity per la verifica on-chain della prova

Al momento, l'unica possibilità che viene offerta per utilizzare ZoKrates in modo non nativo è la libreria zokrates-js, che permette di usufruire delle funzionalità di questo strumento anche in ambiente Javascript (è anche disponibile l'installazione via npm). Esiste anche un plugin di ZoKrates per RemixIDE, un editor web molto utilizzato da chi sviluppa su Ethereum, che permette di utilizzare le funzionalità della suite ZoKrates all'interno di un ambiente di sviluppo integrato.

È bene ricordare che la complessità del programma ZoKrates determina il burden da sostenere dal punto di vista computazionale e di dimensioni del circuito compilato, delle due chiavi e quindi dello smart contract (la verification key è inclusa nel contratto). Il limite massimo per uno smart contract su Ethereum è di 24KB, perciò questo aspetto non è da sottovalutare, sebbene siano possibili dei workaround per aggirare il problema del deploy (es. upload della chiave dopo il deploy del contratto); in fase di programmazione è quindi consigliabile fare delle scelte tecniche oculate che riducano al minimo la complessità del circuito.

2.3.1.3 MiMC Hash

MiMC (Minimal Multiplicative Complexity) è una famiglia di funzioni di hash progettate specificatamente per essere utilizzate in sistemi zk-SNARK. Esse infatti sono state costruite cercando di minimizzare la complessità moltiplicativa, che ha un impatto importante sulle performance di questo genere di applicazioni. Per sottolineare l'importanza di questo aspetto si può pensare ad esempio di generare una merkle proof con ZoKrates: utilizzando una funzione comunemente nota come SHA256 per costruire l'albero si assiste a una crescita quasi logaritmica dei tempi di generazione della prova all'aumentare del numero di foglie, oltre a un incremento importante delle dimensioni di proving e verification key e di conseguenza di quelle del contratto di verifica. Questo rende di fatto impraticabile utilizzare alberi anche solo con una decina di foglie. MiMC invece gestisce in maniera efficiente alberi molto più grandi, e più in generale offre delle performance che permettono di utilizzare le ZKP in scenari reali e non puramente teorici.

2.3.2 Proposte iniziali

2.3.2.1 Introduzione

La definizione del Proof of Concept è partita da due possibili tematiche di base legate alla tecnologia blockchain, la gestione di dati off-chain tramite oracoli e la certificazione dei dati utilizzati per calcolare i token da assegnare agli utenti. Dopo un'attenta discussione tra le parti si è deciso che, per il contesto applicativo descritto da ENEA, fosse più significativo approfondire la seconda proposta in quanto rende trasparente il processo di assegnazione dei tokens e dunque, dal punto di vista dell'utente, costituisce realmente un valore aggiunto. A questo punto, il Politecnico ha presentato ad ENEA tre ipotesi di PoC incentrate sul tema prescelto.

2.3.2.2 Proposta 1

Cosa viene richiesto a SINAPSI?

In questo scenario SINAPSI aggiunge al dispositivo utente la capacità di firmare i dati utilizzando una coppia di chiavi privata/pubblica, che può essere generata direttamente dal device (eventualmente l'utente può chiedere di rigenerarle) oppure impostata dall'utente attraverso la sua dashboard di controllo.

Prima fase

Prima di inviare i dati di consumo, il dispositivo li firma utilizzando la chiave privata e aggiunge il risultato al messaggio; poiché, come spiegato di seguito, si vuole fare uso delle Zero-Knowledge Proofs (ZKPs) per certificare il calcolo delle ricompense, in questa fase è opportuno scegliere un algoritmo per la firma che sia facilmente utilizzabile nel processo di generazione di queste prove (es. EdDSA). Nel caso di ZoKrates, ad esempio, viene messa a disposizione degli sviluppatori una libreria standard che contiene solo alcuni tipi di firma; se si sceglie di utilizzare questo tool, è consigliabile utilizzare gli algoritmi già implementati in modo da rendere più agevole lo sviluppo.

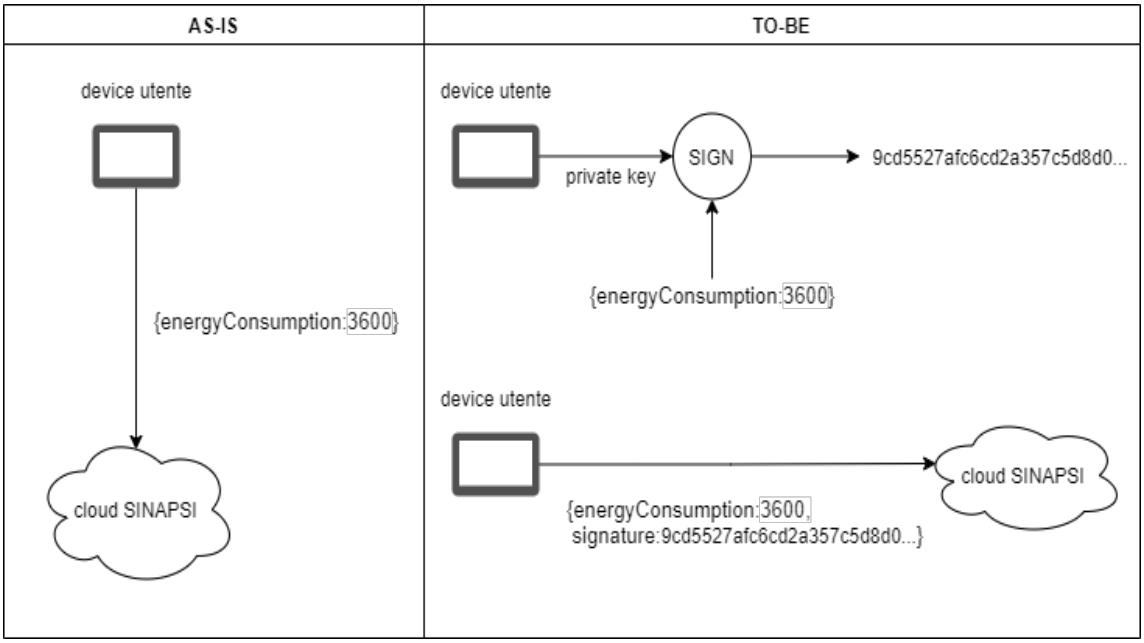

Figure 4: Segmento device utente-cloud SINAPSI nelle configurazioni AS-IS e TO-BE.

Zero-Knowledge Proof

Le Zero-Knowledge Proofs sono prove crittografiche che possono essere utilizzate da un soggetto per dimostrare ad un altro soggetto che un'affermazione è vera, senza rivelare nient'altro oltre alla veridicità della stessa. Attraverso le ZKPs è possibile effettuare una selective disclosure su un determinato set di dati, questo significa che è possibile decidere quali informazioni rendere pubbliche e quali mantenere private; nel caso in esame, inoltre, possono essere utilizzate anche per certificare il calcolo che è stato eseguito per determinare i token da assegnare all'utente.
ENEA, una volta ottenuti i dati del messaggio e la firma, può generare una ZKP che dimostra:

- che il calcolo è stato eseguito seguendo il procedimento corretto;
- che i dati di consumo utilizzati nel calcolo sono quelli firmati dal dispositivo utente.

Una prova con queste caratteristiche può essere creata utilizzando ZoKrates, soppesando bene le scelte implementative in fase di costruzione del circuito (cioè il programma scritto in ZoKrates contenuto nel file con estensione .zok) in modo da limitare il più possibile lo sforzo computazionale necessario a generare la prova. Il circuito deve ricevere come input pubblici, ossia informazioni derivabili dalla prova, la firma, la chiave pubblica del dispositivo, i parametri dell'algoritmo e il numero di token assegnati; come unico input privato, invece, i dati di consumo dell'utente. Il programma è costituito da due parti: una in cui viene verificata la firma, e una in cui viene rieseguito il calcolo delle ricompense a partire dai dati di consumo, controllando anche che il risultato coincida con il numero di token assegnati all'utente. A partire dal circuito è possibile generare automaticamente lo smart contract (scritto in Solidity) per la verifica delle prove, che quindi avviene direttamente on-chain; utilizzando questo contratto è possibile costruire un'applicazione (per esempio in Javascript con le librerie web3.js3 o ethers.js4 o in Python con web3.py5) per permettere all'utente di verificare le ZKPs che riceve da ENEA in autonomia.

2.3.2.3 Proposta 2

Cosa viene richiesto a SINAPSI?

SINAPSI, mentre raccoglie i dati di consumo di un determinato utente, costruisce ad intervalli di tempo regolari un merkle tree e notarizza il root su blockchain.

Prima fase

SINAPSI, ad intervalli di tempo regolari, genera un merkle tree a partire dai dati di consumo raccolti e fa commitment notarizzando il suo root su blockchain, per esempio tramite emit di un evento in uno smart contract. Dal punto di vista della notarizzazione un approccio di questo tipo permette di risparmiare notevolmente in termini di gas rispetto al salvataggio del dato all'interno dello smart contract. Quando ENEA fa una chiamata alle API di SINAPSI per leggere i dati di consumo di un utente, costruisce lo stesso merkle tree da cui ricava il root e le merkle proofs; queste informazioni vengono utilizzate nella fase successiva per generare la ZKP.

Zero-Knowledge Proof

ENEA costruisce una ZKP passando come input pubblici il merkle tree, le merkle proofs, i parametri dell'algoritmo e il numero di token assegnati e come input privati i dati di consumo dell'utente. Questa prova certifica il calcolo effettuato per assegnare i token e dimostra che i dati di consumo utilizzati in input sono contenuti nel merkle root che è stato notarizzato.
Analogamente a come avviene nella Proposta 1, una prova di questo tipo può essere prodotta tramite ZoKrates.

2.3.2.4 Proposta 3

Cosa viene richiesto a SINAPSI?

In questa ultima proposta si ipotizza una soluzione molto simile alla Proposta 1, con la differenza che in questo caso il dispositivo utente non è in grado di produrre una firma.
Per ovviare a questo problema si delega il compito di certificare i dati a SINAPSI, che quindi li firma in prima persona prima di inviarli ad ENEA. Questa scelta è giustificata dal fatto che in questo contesto SINAPSI rappresenta un attore trusted, per cui la sua firma può essere considerata una garanzia sufficiente.

Prima fase

Quando ENEA richiede i dati di consumo di un utente inviando delle requests alle sue API, SINAPSI firma le risposte prima di spedirle (es. tramite un proxy che intercetta le risposte in uscita e aggiunge la firma).

Zero-Knowledge Proof

ENEA, una volta ricevuto i dati di consumo e la firma di SINAPSI, genera una ZKP che prova:

- che il calcolo è stato eseguito seguendo il procedimento corretto;
- che i dati di consumo utilizzati nel calcolo sono quelli firmati da SINAPSI.

La prova può essere creata con ZoKrates in modo del tutto analogo alla Proposta 1, sostituendo tra gli input pubblici la chiave pubblica del dispositivo con la chiave pubblica di SINAPSI.

2.3.3 Evoluzione del progetto

2.3.3.1 Scenario definitivo

Dopo aver constatato la difficoltà di ottenere da SINAPSI l'implementazione delle soluzioni necessarie alla realizzazione di una delle proposte elencate in precedenza, si è deciso di puntare al PoC descritto nella Proposta 3, astraendo però dalla certificazione dei dati da parte di SINAPSI (pur considerando possibili problemi implementativi, per esempio di performance). Lo scenario finale che è stato delineato prevede la certificazione del calcolo effettuato per l'assegnazione dei tokens attraverso l'utilizzo di Zero-Knowledge Proofs; in sede di *minting* di nuovi tokens ENEA presenta allo Smart Contract una ZKP che certifica il calcolo effettuato, la quale viene verificata e notarizzata direttamente on-chain. Il codice ZoKrates utilizzato per generare le prove e il contratto verificatore vengono resi pubblici, in questo modo ENEA fornisce delle garanzie sufficienti a convincere chiunque del fatto che il conteggio sia stato effettuato nel modo corretto; l'utente inoltre è in grado di verificare che siano stati utilizzati i dati raccolti dal suo contatore senza alcuna alterazione, in quanto il merkle root dei consumi è un dato pubblico ricavabile direttamente dalla prova.

2.3.3.2 Considerazioni per l'implementazione

ZoKrates è il tool che è stato scelto per agevolare la creazione delle Zero-Knowledge Proofs. Questo strumento mette a disposizione un linguaggio di programmazione ad alto livello, che viene poi compilato per essere tradotto in un circuito matematico; per generare una prova, è necessario scrivere un programma con questo linguaggio, in cui dovranno essere distinti con precisione gli *input privati* da quelli *pubblici*. Gli ingressi privati sono i dati che non dovranno essere ricavabili dalla prova, mentre quelli pubblici verranno inseriti "in chiaro" nella ZKP; a livello di programmazione, questa distinzione si effettua semplicemente caratterizzando o meno le variabili di ingresso del programma con la parola chiave **private**.

Come già ricordato in precedenza, nella scrittura di un programma ZoKrates in generale è sempre consigliabile soppesare con attenzione le scelte implementative, per evitare di incrementare in modo consistente la complessità del circuito e quindi incorrere in costi computazionali significativi quando si generano le prove. Nel caso in esame, per esempio, il fatto che il circuito debba ricostruire il Merkle root dei consumi impone necessariamente una riflessione sull'hash da utilizzare per generare l'albero. Funzioni come SHA-256 si rivelano poco adatte allo scopo, infatti appesantiscono di molto il circuito anche in presenza di alberi con poche foglie; soluzioni come MiMC invece forniscono prestazioni decisamente migliori.

Un altro punto da considerare con attenzione è la blockchain su cui si vuole implementare il sistema. Operazioni come la verifica di Zero-Knowledge Proofs direttamente on-chain implicano un carico computazionale non trascurabile che, soprattutto se effettuate frequentemente, potrebbero condurre a un incremento significativo dei costi di mantenimento dell'applicazione. Da un punto di vista meramente economico potrebbe risultare vantaggioso utilizzare una piattaforma *permissioned*, ma questa soluzione offre meno trasparenza rispetto all'implementazione su una blockchain *permissionless*. Anche tra i network *permissionless* la situazione è eterogenea: a seconda del caso in esame ci si può imbattere in network fees più o meno alte, e possono essere presenti o meno delle soluzioni di scaling che permettono di ridurre i costi operativi.

2.3.3.3 Struttura della prova

La prova generata da ZoKrates si configura come un file .json, che al suo interno contiene sempre queste quattro chiavi: *a,b,c e inputs*. Mentre a,b e c rappresentano la prova vera a propria, inputs è una lista che

contiene i dati caratterizzati come pubblici durante la scrittura del programma; in questo caso, al suo interno l'utente può trovare il merkle root utilizzato in sede di calcolo e confrontarlo con i propri consumi.

2.3.3.4 Flusso operativo

All'interno dello scenario descritto in precedenza, si prevede il seguente flusso operativo:

1. ENEA crea il circuito ZoKrates ed effettua il deploy del contratto Verifier
    a. `zokrates compile` → compila il circuito
    b. `zokrates setup` → crea proving e verification key
    c. `zokrates export-verifier` → genera contratto Verifier utilizzando la verification key
    d. deploy contratto Verifier
2. ENEA, dopo aver calcolato i token da assegnare a un certo utente, crea la relativa prova
    a. `zokrates compute-witness` → genera un testimone per il circuito a fronte degli input che gli sono stati passati (consumi utente, parametri algoritmo, ecc)
    b. `zokrates generate-proof` → genera la prova utilizzando il witness generato in precedenza e la proving key
3. ENEA chiede allo smart contract ERC-20 di mintare i token calcolati, fornendo la ZKP generata in precedenza
    a. Lo smart contract ERC-20 verifica la prova tramite il contratto Verifier e, se l'operazione va a buon fine, salva la prova on-chain e minta i token richiesti

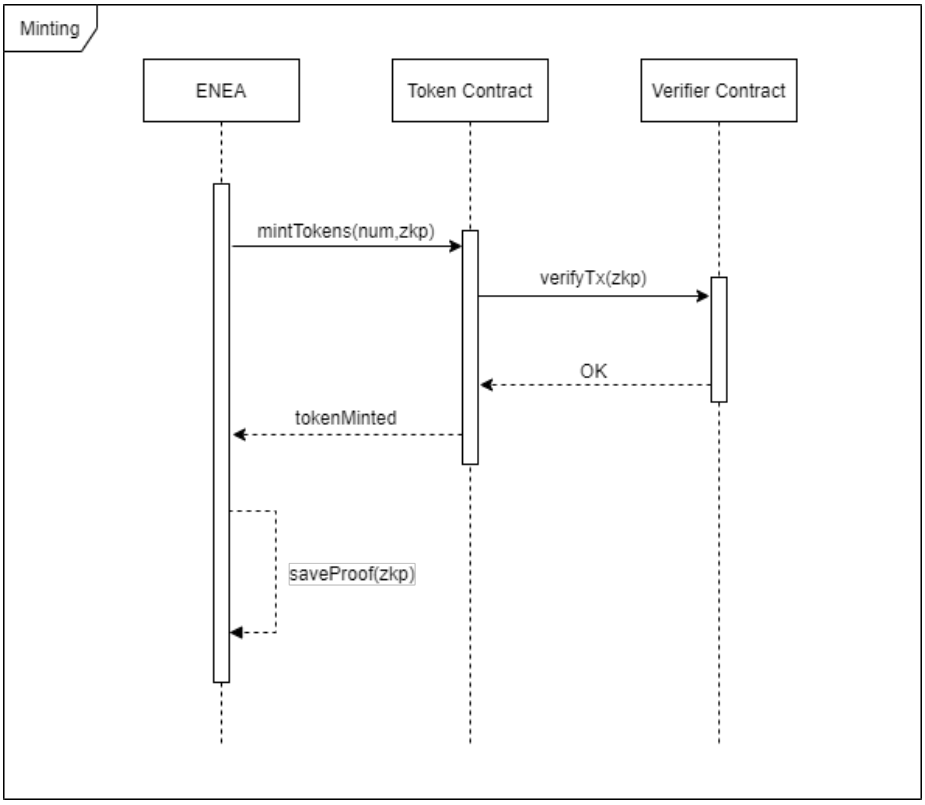

Figure 5: Sequence Diagram del processo di minting di nuovi tokens.

2.3.3.5 Casi d'uso di esempio

Attori
ENEA, Alice (utente)

Minting di nuovi tokens
ENEA effettua il calcolo per determinare quanti token devono essere assegnati ad Alice in base ai suoi consumi, quindi produce una ZKP tramite ZoKrates passando in ingresso al circuito tutti i dati necessari a generare la prova. A questo punto, invia una transazione al Token Contract per *mintare* i token determinati in precedenza includendo anche la ZKP appena prodotta. Il Token Contract interagisce con il Verifier Contract per verificare la prova, quindi *minta* i nuovi token e li assegna ad Alice.

Verifica consumi utilizzati nella determinazione dei tokens
Alice, dopo aver ricevuto x tokens, vuole verificare che ENEA nel calcolo abbia effettivamente utilizzato i suoi consumi. Per fare questo calcola off-chain il Merkle root dei propri dati, quindi ottiene dal Token Contract la ZKP che è stata verificata in fase di minting e controlla che la chiave "inputs" contenga effettivamente lo stesso root.
Una circostanza notevole di questo flusso è che *chiunque* si può convincere del fatto che *tutti* i token siano stati creati in modo corretto, cioè applicando le stesse regole generative/condizioni a partire da certi dati in ingresso, *senza* conoscere direttamente i dati, ma soltanto un loro particolare *hash*.

3 Conclusioni

Lo studio iniziale sugli oracoli e sul rapporto tra blockchain e Big Data è stato affiancato da un'attenta analisi dell'ambito applicativo considerato, e si è giunti alla conclusione che all'interno di questo scenario fosse più significativo certificare il calcolo effettuato per l'assegnazione dei tokens piuttosto che notarizzare i dati di consumo degli utenti. L'ipotesi di PoC a cui si è giunti permette all'utente di verificare che i tokens ricevuti siano stati calcolati correttamente, e nel contempo tutela la sua privacy in quanto i dati di consumo restano noti solamente a se stesso e ad ENEA. Nello scenario che è stato delineato, infatti, ENEA è costretto a generare e caricare on-chain una ZKP ogni volta che minta dei nuovi tokens: da un lato, questa operazione offre agli utenti delle garanzie concrete sul processo di assegnazione dei tokens, dall'altro, non mette a rischio la privacy dell'utente, in quanto l'unico dato ricavabile dalla prova risulta essere il Merkle root dei consumi (che non è un dato riservato).

4 Riferimenti bibliografici

4.1 Paragrafo 2.1

Oracoli blockchain

https://betterprogramming.pub/what-is-a-blockchain-oracle-f5ccab8dbd72
https://medium.com/fabric-ventures/decentralised-oracles-a-comprehensive-overview-d3168b9a8841
https://coinmarketcap.com/alexandria/article/oracles-the-all-seeing-eye-that-guides-crypto-networks
https://arxiv.org/pdf/2004.07140.pdf
https://www.researchgate.net/publication/346063971_Tunnelling_Trust_into_the_Blockchain_a_Merkle_Based_Proof_System_for_Structured_Documents
https://www.researchgate.net/publication/340662783_A_Study_of_Blockchain_Oracles

TownCrier

https://town-crier.org/dev/
https://blog.chain.link/town-crier-and-chainlink/

Oraclize

https://docs.provable.xyz/
https://medium.com/coinmonks/using-apis-in-your-ethereum-smart-contract-with-oraclize-95656434292e

Augur

https://kriptomat.io/augur/
https://atomarsexchange.medium.com/general-overview-of-augur-rep-ca242124b5d1

Kleros

https://kleros.io/assets/whitepaper.pdf
https://medium.com/kleros/kleros-frequently-asked-questions-about-peer-to-peer-justice-5a921cb76abe

Chainlink

https://link.smartcontract.com/whitepaper
ChainLink documentation, https://docs.chain.link/
https://blokt.com/guides/chainlink

Band Protocol

https://research.binance.com/en/projects/band-protocol
https://medium.com/coinmonks/what-is-band-protocol-band-375063aeab3e

4.2 Paragrafo 2.2

Big Data

https://suryagutta.medium.com/the-5-vs-of-big-data-2758bfcc51d
http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity- and-Variety.pdf
https://www.zdnet.com/article/volume-velocity-and-variety-understanding-the-three-vs-of-big-data/
https://blog.osservatori.net/it_it/le-5v-dei-big-data

Schemi commit-reveal

https://medium.com/gitcoin/commit-reveal-scheme-on-ethereum-25d1d1a25428
https://medium.com/swlh/exploring-commit-reveal-schemes-on-ethereum-c4ff5a777db8
http://crypto.cs.mcgill.ca/~crepeau/PDF/Commit.pdf
https://docs.soliditylang.org/en/develop/solidity-by-example.html#blind-auction
https://homepages.cwi.nl/~schaffne/courses/crypto/2014/papers/ComZK08.pdf

Merkle tree

https://www.emsec.ruhr-uni-bochum.de/media/crypto/attachments/files/2011/04/becker_1.pdf
https://www.investopedia.com/terms/m/merkle-tree.asp#:~:text=A%20Merkle%20tree%20is%20a,as%20%22binary%20hash%20trees.%22

Zero Knowledge Proof

http://pages.cs.wisc.edu/~mkowalcz/628.pdf
https://www.bennetyee.org/ucsd-pages/ZKP.html
https://academy.binance.com/en/glossary/zero-knowledge-proofs
https://academy.binance.com/it/articles/zk-snarks-and-zk-starks-explained
https://www.zeroknowledgeblog.com/index.php/zk-snarks

5 Abbreviazioni ed acronimi

- ZKP: Zero-Knowledge Proof
- PoC: Proof of Concept
- CLI: Command Line Interface

6 Gruppo di lavoro

- *nome*: **Francesco Bruschi**
  titoli di studio:
  - Diploma di dottorato in Information Engineering, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy, final evaluation A – cum Laude (esame finale 18/05/2004)
  - Laurea (Vecchio Ordinamento) in Ingegneria Elettronica, 2000, Politecnico di Milano, Milano, Italia

  posizione:
  - Assistant Professor presso il Dipartimento di Elettronica, Informazione e Bioingegneria del Politecnico di Milano
  - Direttore dell'Osservatorio Blockchain and Distributed Ledger del Politecnico di Milano

  esperienza professionale:
  - Francesco è ricercatore di ruolo MIUR e docente presso il Politecnico di Milano, oltre che direttore dell'Osservatorio Blockchain and Distributed Ledger del Politecnico di Milano. Ha partecipato a numerosi progetti di ricerca e comitati scientifici internazionali, ed è autore di molte pubblicazioni scientifiche

  pubblicazioni rilevanti:
  - title: "A Decentralized System for Fair Token Distribution and Seamless Users Onboarding"
    authors: F. Bruschi, M. Tumiati, V. Rana, M. Bianchi, D. Sciuto
    2020 IEEE Symposium on Computers and Communications (ISCC), 1-6 2020
  - title: "Tunnelling Trust into the Blockchain: a Merkle Based Proof System for Structured Document"
    authors: F. Bruschi, V. Rana, A. Pagani, D. Sciuto
    IEEE Access 2020
  - title: "Acknowledging Value of Personal Information: a Privacy Aware Data Market for Health and Social Research"
    authors: F. Bruschi, V. Rana, A. Pagani, D. Sciuto
    DLT@ ITASEC 2020
  - title: "Le applicazioni delle nuove tecnologie: criptovalute, blockchain e smart contract"
    author: F. Bruschi
    year: 2020
  - title: "Proof system and method for structured documents"
    authors: F. Bruschi, V. Rana
    year: 2020
  - title: "Mine with it or sell it: the superhashing power dilemma"
    authors: F. Bruschi, V. Rana, L. Gentile, D. Sciuto
    ACM SIGMETRICS Performance Evaluation Review 46 (3), 127-130 4 2019

- *nome*: **Vincenzo Rana**
  posizione:
  - Docente del Politecnico di Milano e del MIP
  - Ricercatore dell'Osservatorio Blockchain & Distributed Ledger del Politecnico di Milano
  - CEO di Knobs srl

  titoli di studio:
  - B.Sc. in Ingegneria Informatica, Luglio 2004
  - M.Sc. in Ingegneria Informatica, Ottobre 2006
  - Ph. D. in Ingegneria dell'Informazione, Marzo 2010

  esperienza professionale:
  - Vincenzo Rana è docente del Politecnico di Milano e del MIP, oltre ad essere ricercatore dell'Osservatorio Blockchain & Distributed Ledger del Politecnico di Milano. Le sue principali aree di ricerca spaziano dalle tecnologie blockchain all'ottimizzazione di algoritmi computazionalmente intensivi per l'analisi dei dati e alla progettazione di architetture hardware efficienti e ad alte prestazioni. Vincenzo Rana è stato autore di 60 articoli su riviste e atti di conferenza internazionali e 6 capitoli di libro.

  pubblicazioni rilevanti:
  - title: "A Decentralized System for Fair Token Distribution and Seamless Users Onboarding"
    authors: F. Bruschi, M. Tumiati, V. Rana, M. Bianchi, D. Sciuto
    2020 IEEE Symposium on Computers and Communications (ISCC), 1-6 2020
  - title: "Tunnelling Trust into the Blockchain: a Merkle Based Proof System for Structured Document"
    authors: F. Bruschi, V. Rana, A. Pagani, D. Sciuto
    IEEE Access 2020
  - title: "Acknowledging Value of Personal Information: a Privacy Aware Data Market for Health and Social Research"
    authors: F. Bruschi, V. Rana, A. Pagani, D. Sciuto
    DLT@ ITASEC 2020
  - title: "Proof system and method for structured documents"
    authors: F. Bruschi, V. Rana
    year: 2020
  - title: "Mine with it or sell it: the superhashing power dilemma"
    authors: F. Bruschi, V. Rana, L. Gentile, D. Sciuto
    ACM SIGMETRICS Performance Evaluation Review 46 (3), 127-130 4 2019

- *nome*: **Davide Ghezzi**
  posizione:
  - Assegnista di ricerca presso l'Osservatorio Blockchain and Distributed Ledger del Politecnico di Milano

  titoli di studio:
  - B.Sc. in Ingegneria Gestionale, 23 Settembre 2017
  - M.Sc. in Management Engineering, 15 Dicembre 2020

  esperienza professionale:
  - Davide è un assegnista di ricerca presso l'Osservatorio Blockchain and Distributed Ledger del Politecnico di Milano. Le sue principali aree di ricerca riguardano: trend tecnici e di business inerenti a blockchain e DLT, analisi e approfondimento di alcuni aspetti tecnici collegati a queste tecnologie (smart contracts, dapps, tokens, oracoli) e analisi del livello di adozione della blockchain tra le aziende italiane e mondiali

pubblicazioni rilevanti:

- M.Sc. Thesis
  title: "The Role of Blockchain and Distributed Ledger Technologies in Business Innovation: a Comprehensive Analysis of the International Blockchain Startup Ecosystem"
  author: Davide Ghezzi supervisor: Alessandro Perego co-supervisor: Valeria Portale, Jacopo Fracassi, Giacomo Vella
- Research report
  title: "Blockchain: the hype is over, get ready for ecosystems"
  published at: Osservatori.net
  date: January 2021
- Research report
  title: "Blockchain & Distributed Ledger nel 2020: i progetti nel mondo e i principali casi d'uso"
  published at: Osservatori.net
  date: March 2021
- Research report
  title: "Come ottenere privacy e controllo dei dati all'interno delle piattaforme Blockchain e Distributed Ledger"
  published at: Osservatori.net
  date: March 2021

- *nome*: **Tommaso Paulon**
  posizione:
  - Assegnista di ricerca presso il Dipartimento di Elettronica, Informazione e Bioingegneria del Politecnico di Milano
  - Collaboratore esterno presso BCode srl

  titoli di studio:
  - B.Sc. in Ingegneria Informatica, 24 Febbraio 2017
  - M.Sc. in Computer Science and Engineering, 15 Dicembre 2020

  esperienza professionale:
  - Tommaso in precedenza ha lavorato per tre anni come SAP ABAP & Fiori developer presso Acqua Minerale San Benedetto S.p.A. Attualmente ricopre la posizione di assegnista di ricerca presso il DEIB del Politecnico di Milano; le sue principali aree di ricerca sono sistemi privacy-preserving per applicazioni decentralizzate su blockchain e sistemi basati su Zero Knowledge Proofs

  pubblicazioni rilevanti:
  - M.Sc. Thesis
    title: "A Self-Sovereign Identifiability Solution for Smart Contracts Using Zero-Knowledge Proofs"
    author: Tommaso Paulon
    supervisor: Francesco Bruschi
    co-supervisor: Vincenzo Rana