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

Ministero della Transizione Ecologica

Ricerca di Sistema elettrico

Sviluppo di un dimostratore per la generazione e la verifica di token usando dati esterni alla blockchain

Report Tecnico

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

POLITECNICO MILANO 1863

SCHOOL OF MANAGEMENT

DIPARTIMENTO DI ELETTRONICA, INFORMAZIONE E BIOINGEGNERIA

Report RdS/PTR(2021)/051

SVILUPPO DI UN DIMOSTRATORE PER LA GENERAZIONE E LA VERIFICA DI TOKEN USANDO DATI ESTERNI ALLA BLOCKCHAIN

F. Bruschi, V. Rana, D. Ghezzi, T. Paulon – Dipartimento di Ingegneria Informatica Politecnico di Milano

Dicembre 2021

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero della Transizione Ecologica - ENEA

Piano Triennale di Realizzazione 2019-2021 – III annualità

Obiettivo: Tecnologie

Progetto: Tecnologie per la penetrazione efficiente del vettore elettrico negli usi finali

Work package: Local Energy District

Linea di attività 1.64: Integrazione dell'applicazione e dell'architettura dati 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 "*Supporto allo sviluppo di un'applicazione per la valorizzazione del virtuosismo energetico*"

Responsabile scientifico ENEA: Nicola Gessa

Responsabile scientifico Politecnico di Milano, Dipartimento di Ingegneria Informatica: Vincenzo Rana

Indice

Sommario

L'attività descritta in questo report riguarda la progettazione e realizzazione di un sistema decentralizzato di generazione di ricompense ed incentivi per comportamenti energeticamente virtuosi. Le ricompense sono erogate sotto forma di token, e la correttezza dell'assegnamento è dimostrata tramite prove crittografiche pubblicamente verificabili.

Il presente documento è diviso in due parti principali: la prima parte parte analizza più ad alto livello il Proof of Concept (PoC) che è stato realizzato nell'ambito della collaborazione tra ENEA e il Politecnico di Milano, cercando di evidenziare non soltanto gli elementi innovativi e i vantaggi che un sistema di questo tipo introduce ma anche le assunzioni su cui si basa ed eventuali limiti; la seconda parte invece descrive nel dettaglio tutti gli aspetti più tecnici e l'infrastruttura del sistema.

PARTE A

1. Introduzione

Il PoC descritto in questo documento ha lo scopo di realizzare un sistema basato su blockchain per l'assegnazione certificata e verificabile di tokens come ricompensa per il virtuosismo energetico degli utenti. In questa prima parte vengono discussi i vantaggi che un'applicazione di questo tipo può introdurre, le assunzioni su cui si basa e i limiti di questa soluzione.

Dal punto di vista tecnico, con "virtuosismo energetico" possiamo intendere una qualsiasi funzione del consumo di un utente in un certo periodo, come per esempio il fatto che il consumo si sia attestato sotto una certa soglia. Se consideriamo con c(t) la funzione che descrive il consumo di un utente nel tempo, la virtuosità nell'intervallo t0-t1 può essere definita come un funzionale:

V : c, t0, t1 -> R

Un esempio di funzionale potrebbe essere restituire 1 se il consumo è sotto una certa soglia, 0 altrimenti:

$$1 - H\left(\int_{t0}^{t1} c(t)dt - T\right)$$

dove H è la funzione gradino di Heaviside, t0 e t1 sono i valori che definiscono il periodo considerato, e T è un valore soglia.

2. Requisiti

Il sistema che si intende realizzare, al contrario delle soluzioni più tradizionali, vuole offrire all'utente estrema trasparenza sul processo di assegnazione delle ricompense senza però danneggiare la sua privacy, mantenendo quindi privati i dati di consumo. L'approccio più comune in questo genere di applicazioni è la gestione centralizzata delle rewards, che non offre alcuna garanzia e si presta facilmente a manipolazioni; lo scopo principale di questo dimostratore è proprio superare questo limite, continuando comunque a proteggere le informazioni riservate degli utenti.

ENEA quindi intende realizzare un sistema di ricompense per il virtuosismo energetico che soddisfi i seguenti requisiti:

- consentire a utenti o auditors esterni di verificare la correttezza del calcolo
- preservare la privacy degli utenti tutelando i dati di consumo
- ottimizzare i costi della soluzione

Questi obiettivi possono essere raggiunti grazie all'utilizzo di alcune tecnologie innovative quali tokens e Zero-Knowledge Proofs, che verranno introdotte nel prossimo paragrafo.

3. Tecnologie impiegate

Il termine *token* deriva dall'inglese arcaico e viene comunemente utilizzato per indicare degli oggetti simili a monete, di scarso valore intrinseco, che vengono rilasciati privatamente e per un utilizzo ben preciso (es. gettoni per la lavanderia, sale giochi, ecc.). Con l'avvento della tecnologia blockchain questa parola ha acquisito un nuovo significato, infatti in questo ambito viene utilizzata per descrivere astrazioni basate su blockchain che possono essere possedute e rappresentano degli asset, delle valute oppure il diritto di accesso a determinati luoghi o servizi.

Su una piattaforma come Ethereum i token vengono rappresentati come parti di stato di un'applicazione blockchain; per creare un nuovo token è quindi necessario deployare un nuovo smart contract che gestisce tutti gli aspetti fondamentali, ossia proprietà, trasferimenti e diritti di accesso. In questo contesto quindi essi sono ben distinti dal token nativo della piattaforma, ossia l'Ether, infatti vengono gestiti a livello di smart contracts e non a livello di protocollo come quest'ultimo.

Un token viene detto *fungibile* quando una sua singola unità può essere scambiata con un'altra senza alcuna differenza di prezzo o di funzione; al contrario, i cosiddetti token *non fungibili* o *NFT* rappresentano un asset unico e dunque non sono intercambiabili. Uno smart contract che controlla un determinato token fungibile può essere visto idealmente come una tabella in cui sono indicati gli indirizzi e i relativi saldi; quando un token viene trasferito, il contratto decrementa il saldo relativo al mittente e incrementa il saldo del destinatario.

Il rischio di controparte è il rischio che l'altra parte in una transazione non rispetti i propri obblighi. Alcune transazioni sono soggette a un maggiore rischio di controparte perché sono coinvolti più di due attori; si pensi ad esempio alla vendita di un certificato di deposito di un diamante, in questo caso sono chiamati in causa il venditore, il compratore ma anche il custode del bene. E' bene quindi sottolineare che i token che rappresentano oggetti intrinseci alla blockchain non sono soggetti a rischio di controparte aggiuntivo, in quanto l'applicazione del meccanismo di consenso e il controllo delle chiavi private è equivalente al possesso diretto dell'asset, senza intermediari. Lo stesso non vale invece per i token che rappresentano degli asset estrinseci alla blockchain, perciò per mitigare il rischio di controparte in questi casi è necessario capire chi custodisce l'asset e a quali regole esso è soggetto.

I *token standard* descrivono le specifiche minime per una implementazione; per essere compliant con un determinato standard è sufficiente implementare tutte le funzionalità da esso previste, ma è anche possibile aggiungerne di nuove. Lo scopo primario di questi standard è incrementare l'interoperabilità tra gli smart contracts in modo tale che exchanges, wallets e interfacce utente possano interagire in modo predicibile con ogni contratto che segue la specifica; uno dei più famosi e utilizzati su Ethereum è sicuramente ERC-20, lo standard per i token fungibili. Si riporta di seguito l'interfaccia descritta da questo standard:

Metodi

```
function name() public view returns (string)
function symbol() public view returns (string)
function decimals() public view returns (uint8)
function totalSupply() public view returns (uint256)
function balanceOf(address _owner) public view returns (uint256 balance)
function transfer(address _to, uint256 _value) public returns (bool success)
function transferFrom(address _from, address _to, uint256 _value) public returns (bool success)
function approve(address _spender, uint256 _value) public returns (bool success)
function allowance(address _owner, address _spender) public view returns (uint256 remaining)
```

Eventi

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

Un token può essere quindi descritto come un asset digitale che vive sulla blockchain, e si configura come il candidato ideale per rappresentare la ricompensa erogata da ENEA per premiare comportamenti virtuosi dal punto di vista energetico. Se si utilizza una piattaforma permissionless come Ethereum, infatti, le informazioni sulle movimentazioni e sul saldo di ogni indirizzo sono di pubblico dominio, perciò è garantita la trasparenza totale sull'emissione delle ricompense.

Il fatto che questi dati siano pubblici non esclude però la possibilità che ENEA manipoli i conteggi per erogare un numero minore di token; per questo motivo si è scelto di utilizzare le Zero-Knowledge Proofs a garanzia della correttezza del calcolo. In particolare, in questo dimostratore verrà utilizzato il sottoinsieme chiamato *zk-SNARK* (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge), che 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.

Nel processo di generazione delle ZKPs è opportuno soppesare con attenzione le scelte tecnologiche per evitare di incorrere in un aumento considerevole delle dimensioni della prova e quindi dei costi della soluzione. In particolare, l'impiego di algoritmi di hash detti *ZKP-friendly* incide notevolmente da questo punto di vista, soprattutto rispetto all'utilizzo di hash più tradizionali come SHA256; per questo motivo nel PoC verrà utilizzato l'hash MiMC, particolarmente adatto allo scopo in quanto dotato per costruzione di una bassa complessità moltiplicativa.

In fase di generazione della prova è possibile specificare quali informazioni debbano essere pubbliche e quali debbano restare private; ENEA quindi può produrre una ZKP in cui i dati di consumo dell'utente sono nascosti, ma nel contempo il commitment (che viene confrontato con il merkle root di questi dati) è pubblico. In questo

modo, se il commitment è stato precedentemente notarizzato da un soggetto imparziale, ENEA è in grado di dimostrare che non ha manipolato i dati di consumo utilizzati nel calcolo; se il circuito matematico che genera le prove viene reso pubblico, inoltre, la stessa ZKP dimostra anche che il conteggio è stato effettuato seguendo il procedimento prestabilito.

Infine, il fatto che le prove vengano verificate direttamente on-chain e che l'esito negativo della verifica sia in grado di bloccare il minting delle ricompense escludono comportamenti scorretti anche in questa sezione del sistema; le ZKPs vengono salvate nello smart contract e possono essere lette e verificate da chiunque senza alcuna restrizione.

4. Scenario

Il PoC intende implementare la seguente meccanica:

ENEA effettua un calcolo basato sui consumi quartorari dell'utente in una specifica data per determinare quanti tokens debba ricevere per quella giornata, quindi genera una Zero-Knowledge Proof (ZKP) che certifica il conteggio effettuato in cui è contenuto in chiaro un commitment sui dati dell'utente. A questo punto ENEA invia la ZKP al contratto ENEAToken che la verifica direttamente on-chain e, in caso di esito positivo, la salva, genera i tokens e li invia all'utente.

Nello scenario appena descritto ENEA calcola con cadenza giornaliera i tokens da assegnare all'utente, seguendo questa procedura:

- i dati di consumo quartorari di tutti i dispositivi associati all'utente vengono aggregati a parità di quarto d'ora
- per ogni quarto d'ora viene assegnata una penalità se il consumo aggregato supera il parametro SC (soglia di carico) impostato nell'algoritmo
- viene conteggiato il numero di penalità accumulate dall'utente nei 96 confronti relativi alla giornata considerata
- in base al parametro MT (numero massimo di tokens assegnabili per una giornata) impostato nell'algoritmo e alle penalità accumulate per la giornata viene determinato il numero di tokens da inviare all'utente

```
consumi_aggregati = {}
penalties = 0

for c in consumi_utente:
    consumi_aggregati[c.ora] += c.potenza_consumata #somma consumi a parità di quarto d'ora

for q in consumi_aggregati.keys():
    if consumi_aggregati[q] > SC:   #se il consumo aggregato supera la soglia aggiungi penalità
        penalties += 1

token_ottenuti = round(MT - MT/96*penalties)
```

Figura 1. Pseudocodice algoritmo calcolo tokens

A questo punto ENEA genera una Zero-Knowledge Proof che certifica il conteggio delle penalità appena effettuato; questa prova contiene in chiaro un hash che rappresenta un commitment sui consumi impiegati nel calcolo, e può essere confrontato dall'utente con l'hash dei suoi consumi effettivi. ENEA quindi invia la prova al contratto ENEAToken che la verifica on-chain e, in caso di esito positivo, genera ("minta") i tokens per l'utente; al termine di questa operazione ENEAToken salva la prova che è stata utilizzata, in modo che possa essere consultata anche a posteriori.

A lato dello scenario principale appena descritto (di seguito denominato *minting verificato*) è stata mantenuta la possibilità di mintare i tokens in modo tradizionale, in modo da rendere più evidenti le differenze tra i due processi.

5. Assunzioni

E' doveroso sottolineare che in un'applicazione reale un sistema di questo tipo è efficace solamente se il commitment sui dati dell'utente non può essere manipolato da ENEA; in caso contrario l'impiego stesso di un commitment si rivela inutile e non fornisce alcuna garanzia. Questo strumento si identifica come una sorta di "impronta" dei dati che varia ad ogni minima modifica e che viene idealmente notarizzata durante la fase di acquisizione dei consumi; in seguito ad un minting verificato può essere confrontato con l'hash presente in chiaro nella prova per capire se i dati utilizzati nel conteggio corrispondono ai dati precedentemente notarizzati. L'affidabilità di questo meccanismo dipende da come vengono gestiti il calcolo e la pubblicazione del commitment; in uno scenario reale queste funzioni dovrebbero essere svolte da un attore indipendente e non corruttibile, in modo da garantire l'autenticità e l'integrità del dato che viene confrontato con l'hash presente nella prova.

E' evidente che quindi la gestione del commitment non possa essere affidata direttamente ad ENEA, perché sarebbe in grado di manipolare questo dato invalidando la sua funzione. In questo PoC si assume pertanto che il commitment sia stato generato e pubblicato da un soggetto non influenzabile da ENEA e che coincida con l'hash generato dal backend.

Perché la prova certifichi realmente il calcolo eseguito da ENEA, inoltre, è necessario che il circuito ZoKrates utilizzato per la sua generazione venga reso pubblico.

6. Vantaggi

Nello scenario rappresentato dal PoC l'utente è in grado di verificare in prima persona la corretta assegnazione dei tokens, infatti ad ogni minting verificato può recuperare la prova che è stata salvata dallo smart contract per:

- controllare che l'hash dei dati utilizzati nel calcolo corrisponda al commitment sui suoi dati di consumo per la data considerata
- verificare a sua volta la ZKP (che garantisce la correttezza del calcolo effettuato da ENEA per determinare il numero di penalità da assegnare all'utente nel giorno considerato)

In altre parole, in seguito ad un minting verificato l'utente è sempre in grado di verificare che i consumi utilizzati da ENEA nel calcolo non siano stati alterati e inoltre ha la certezza che ENEA abbia conteggiato le penalità seguendo la procedura prestabilita. Questo processo garantisce quindi che l'assegnazione dei tokens avvenga in modo corretto e trasparente e, sotto le assunzioni dichiarate nel paragrafo precedente, esclude la possibilità di manipolazioni da parte di ENEA.

7. User Stories

7.1 Attori

- ENEA
- Bob → utente

7.2 Minting verificato

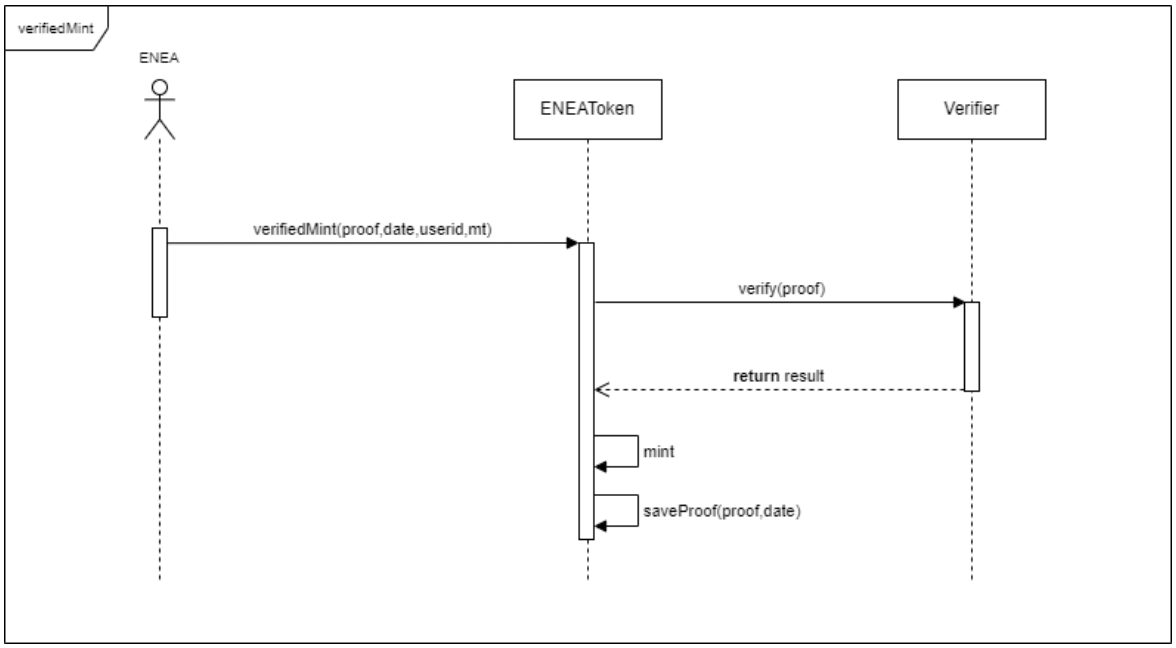

Figura 2. Sequence diagram minting verificato

Stato iniziale

- _balances[Bob]: 0
- verifiedMints[BobID][‘2021-10-05’]: []

Story

ENEA esegue il calcolo sui consumi realizzati da Bob in data 5 ottobre e determina che ha diritto a 3 tokens; a questo punto genera una Zero-Knowledge Proof che certifica il conteggio appena effettuato. Successivamente ENEA chiama il metodo `verifiedMint` del contratto ENEAToken passando la data considerata, l’id di Bob, MT e la prova appena creata; il contratto verifica la prova attraverso una chiamata interna al contratto Verifier, quindi la salva e minta i 3 tokens inviandoli a Bob.

Stato finale

- _balances[Bob]: 3
- verifiedMints[BobID][‘2021-10-05’]: [<proof>]

7.3 Minting non verificato

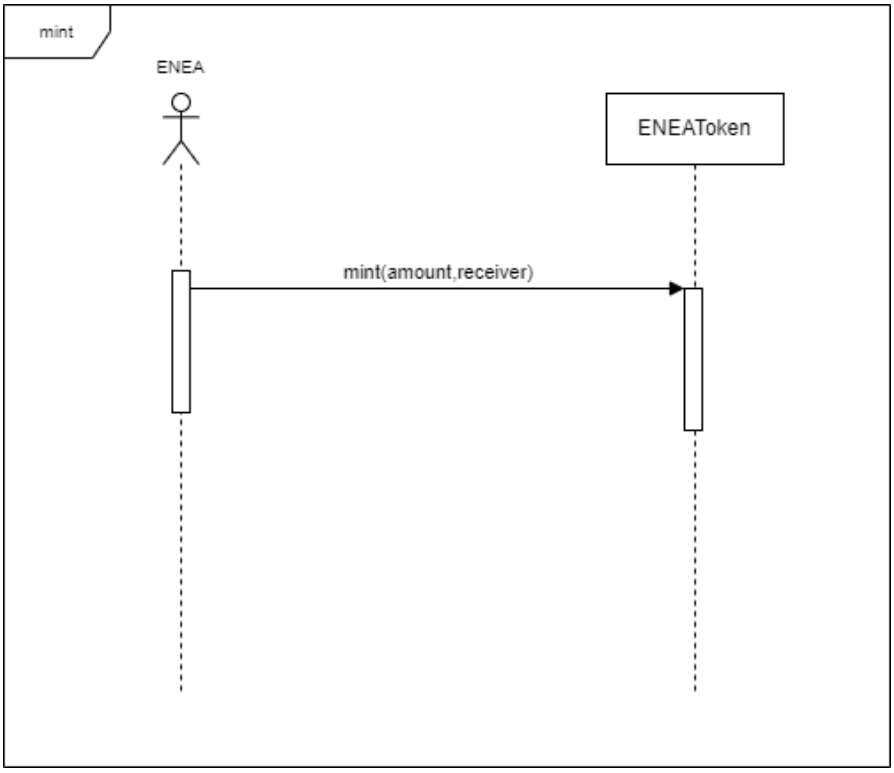

Figura 3. Sequence diagram minting non verificato

Stato iniziale

- _balances[Bob]: 0
- verifiedMints[BobID][‘2021-10-05’]: []

Story

ENEA esegue il calcolo sui consumi realizzati da Bob in data 5 ottobre e determina che ha diritto a 7 tokens, quindi chiama il metodo mintTokens del contratto ENEAToken passando il numero di tokens appena determinato e l’indirizzo che deve riceverli. Il contratto minta i 7 tokens e li trasferisce all’indirizzo di Bob.

Stato finale

- _balances[Bob]: 7
- verifiedMints[BobID][‘2021-10-05’]: []

7.4 Verifica prova a posteriori

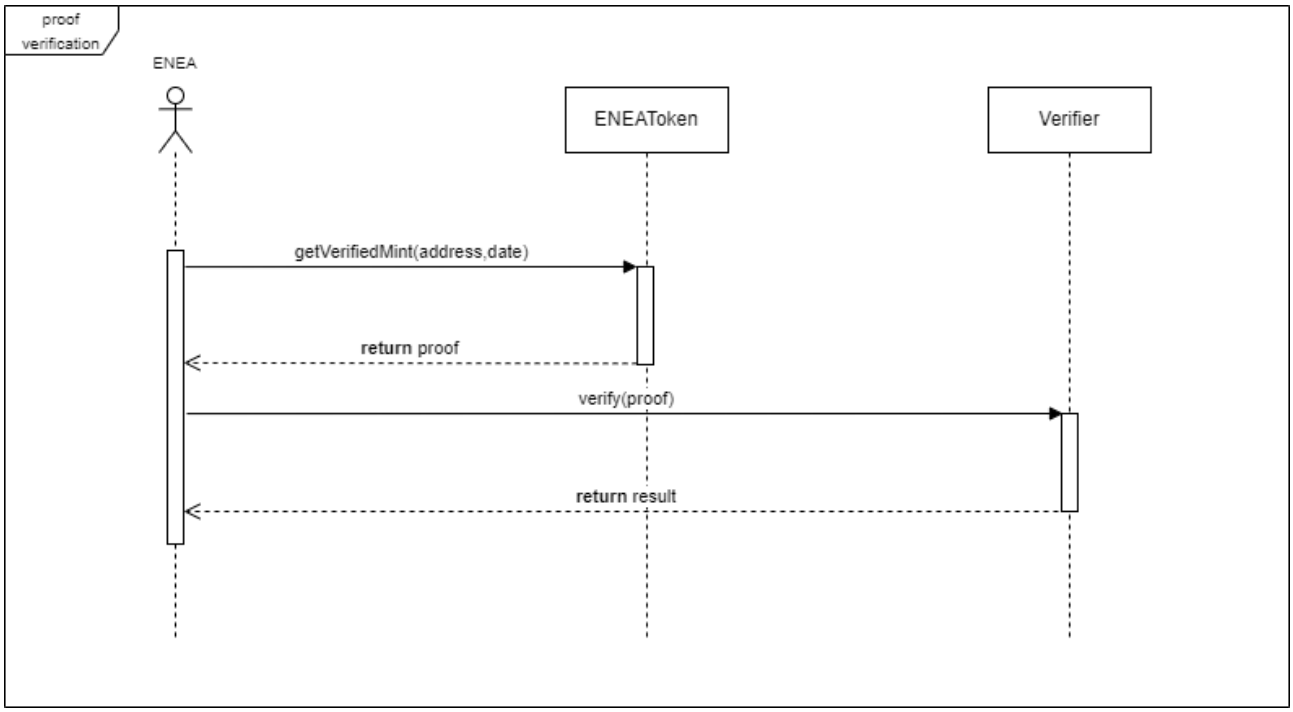

Figura 4. Sequence diagram verifica prova

Stato iniziale

- _balances[BobID]: 3
- verifiedMints[BobID][‘2021-10-05’]: [<proof>]

Story

Bob ha ricevuto 3 tokens per i consumi effettuati in data 5 ottobre, ma vuole verificare il conteggio effettuato da ENEA. Per questo motivo chiama il metodo getVerifiedMint del contratto ENEAToken e identifica la prova <proof> caricata in sede di minting, quindi la invia al metodo verify del contratto Verifier che la verifica e comunica a Bob l’esito positivo dell’operazione.

Stato finale

- _balances[Bob]: 3
- verifiedMints[BobID][‘2021-10-05’]: [<proof>]

PARTE B

1. Introduzione

Docker è un progetto open source nato con lo scopo di automatizzare la distribuzione di applicazioni sotto forma di "contenitori" leggeri, portabili e autosufficienti; a differenza delle macchine virtuali, i container non hanno un sovraccarico elevato e quindi consentono un utilizzo più efficace del sistema e delle risorse sottostanti.

L'infrastruttura del PoC si basa su containers Docker orchestrati tramite Docker Compose, un tool che facilita la gestione e l'esecuzione di applicazioni multicontenitore. Le componenti principali del sistema sono:

- database PostgreSQL
- interfaccia web per gestione db (pgAdmin)
- blockchain (ganache-cli)
- backend (Flask)
- block explorer (alethio)

In questa parte del documento verranno descritti tutti gli elementi che compongono l'infrastruttura e le tecnologie utilizzate nella realizzazione del prototipo.

2. Infrastruttura

Si è deciso di utilizzare Docker per ridurre al minimo i problemi che possono sorgere durante il setup dell'ambiente di sviluppo; molte delle librerie coinvolte in questo PoC infatti sono particolarmente sensibili alle differenze di versione o di sistema operativo. Docker permette di creare le cosiddette Docker Images, ossia dei file contenenti l'insieme dei dati necessari per eseguire una specifica applicazione; in questo modo semplifica notevolmente la fase di distribuzione del software, in quanto per eseguirle è sufficiente disporre di un server Docker.

2.1 Containers

- **name**: postgre
  image: postgres:12.2
  port: 5432
  descr: database postgreSQL in cui viene importato il dump fornito da ENEA, contenente dati di consumo reali o comunque verosimili

- **name**: pgadmin
  image: dpage/pgadmin4:4.18
  port: 8080
  descr: interfaccia grafica per amministrare più agevolmente il database

- **name**: ganache
  image: custom (immagine base node:alpine)
  port: 8545
  descr: blockchain locale

- **name**: backend
  image: custom (immagine base ubuntu:20.04)
  port: 4001
  descr: server Flask che genera le prove ZoKrates, comunica con db e contratti on-chain e serve le interfacce web

- **name**: enea_block_explorer
  image: alethio/ethereum-lite-explorer
  port: 8081
  descr: block explorer locale

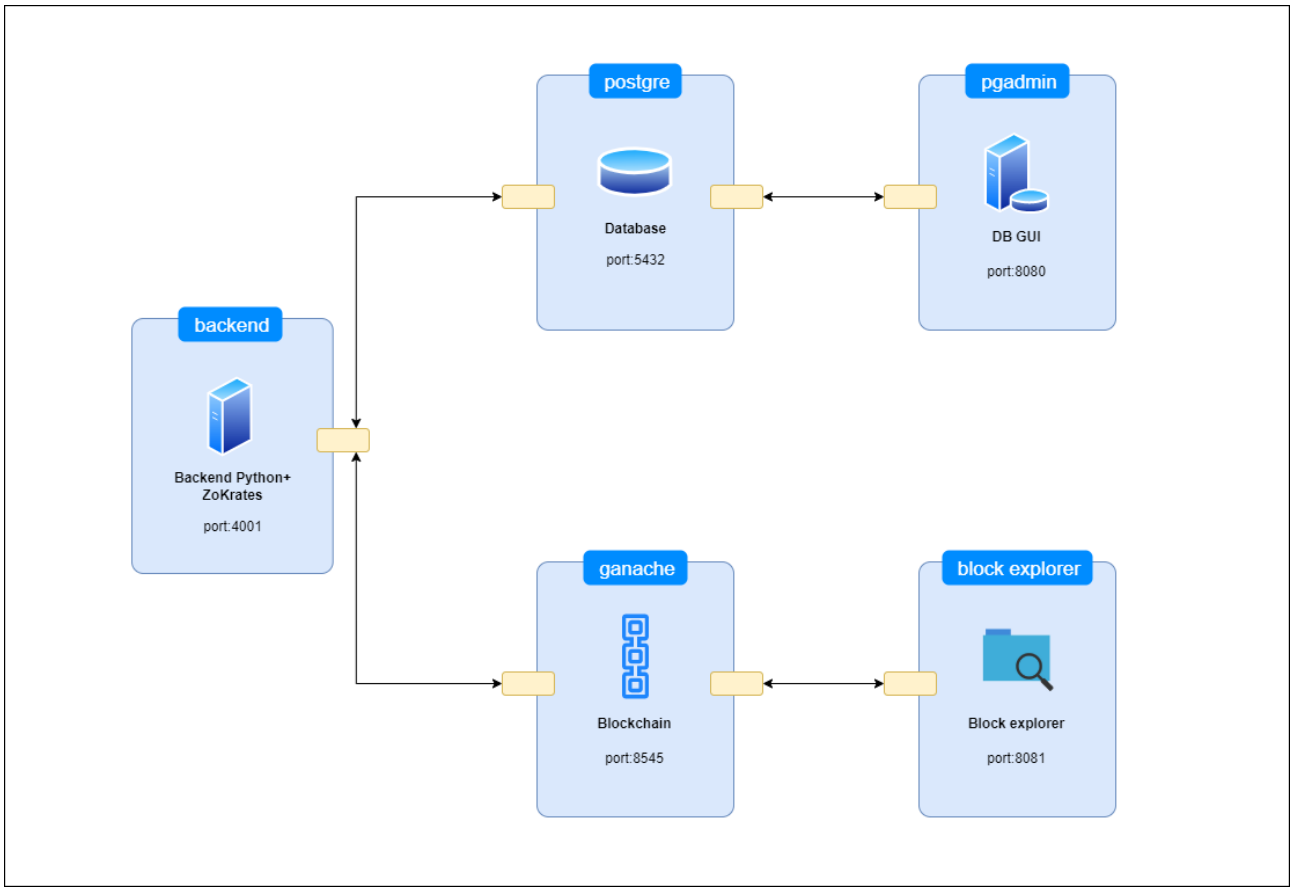

Figura 5. Schema infrastruttura containers

3. Componenti del sistema

3.1 Database

Si è scelto di utilizzare PostgreSQL come database in modo da poter importare più agevolmente il dump fornito da ENEA. Di seguito vengono riportate le variabili d'ambiente fissate di default nel container (sono modificabili da `docker-compose.yaml`):

VARIABILI	VALORE
POSTGRES_DB	postgres
POSTGRES_USER	admin
POSTGRES_PASSWORD	secret
PGDATA	/var/lib/postgresql/data

Tabella 1. Variabili d'ambiente container PostgreSQL

Per poter generare le ZKPs con ZoKrates è necessaria una trasformazione dei dati letti dal database in quanto il tipo fondamentale `field` rappresenta un elemento su campo e assume solamente valori interi positivi compresi tra [0, p - 1] (p è un numero primo molto grande, in questo caso vale 21888242871839275222246405745257275088548364400416034343698204186575808495617).

La trasformazione che viene applicata è la seguente:

- vengono aggregate le rilevazioni quartorarie dei vari device utente per il giorno considerato in modo da ottenere esattamente 96 letture aggregate
- viene sommato un offset pari al dato più negativo a tutte le 96 letture per ottenere solamente valori positivi
- viene moltiplicato ogni dato per un fattore 10^{14} in modo da ottenere solamente valori interi
- viene applicato un arrotondamento per gestire i rari casi in cui a questo punto sia rimasto qualche dato decimale

Con questo procedimento si ottengono quindi 96 numeri interi che rappresentano le letture quartorarie aggregate dell'utente nella giornata considerata; questi dati possono essere rappresentati con il tipo `field` e quindi possono essere utilizzati all'interno del programma ZoKrates.

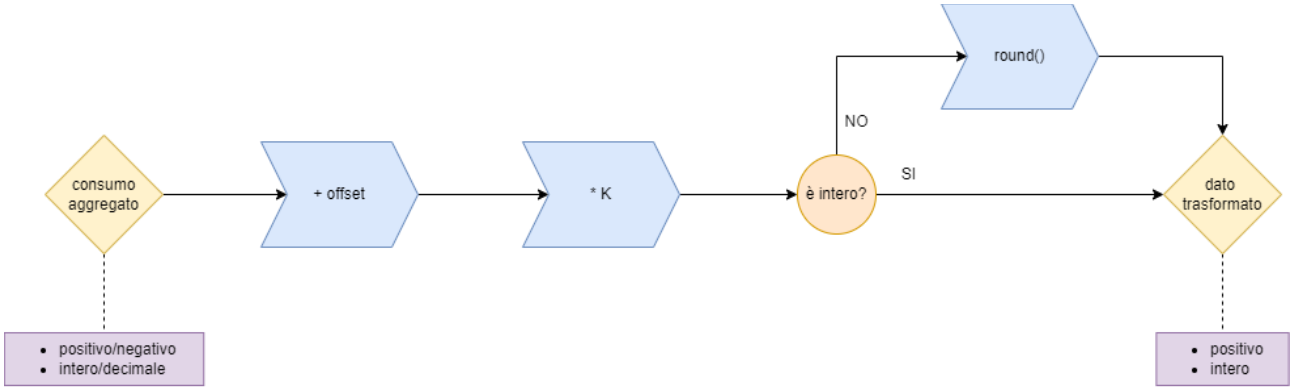

Figura 6. Processo di trasformazione dei dati di consumo

3.2 Block Explorer

Un block explorer è uno strumento utilizzato per ispezionare la blockchain in quanto permette di visualizzare tutte le informazioni pubbliche relative alle transazioni e agli smart contracts. Nell'infrastruttura è incluso un block explorer collegato alla rete locale; di seguito vengono riportate le variabili d'ambiente fissate di default nel container (sono modificabili da `docker-compose.yaml`):

VARIABILI	VALORE
APP_NODE_URL	http://blockchain:8545 (blockchain è un placeholder che assume come valore l'ip del container con l'istanza di ganache-cli)

Tabella 2. Variabili d'ambiente container block explorer alethio

3.3 PgAdmin

PgAdmin è uno strumento che facilita molto la gestione del database, infatti offre un'interfaccia web che permette di interagire con PostgreSQL in modo semplice ed intuitivo. Nonostante non sia strettamente indispensabile per il funzionamento del dimostratore, si è deciso di includere questo tool nell'infrastruttura per velocizzare la visualizzazione dei dati di consumo durante le fasi di sviluppo.

Di seguito vengono riportate le variabili d'ambiente fissate di default nel container (sono modificabili da `docker-compose.yaml`):

VARIABILI	VALORE
PGADMIN_DEFAULT_EMAIL	admin@gmail.com
PGADMIN_DEFAULT_PASSWORD	secret
PGADMIN_LISTEN_PORT	80

Tabella 3. Variabili d'ambiente container pgadmin

3.4 Blockchain

Ganache è un tool molto utilizzato per testare le applicazioni blockchain, infatti crea una rete locale di test che simula il funzionamento del network Ethereum. Questo strumento può essere utilizzato sia tramite interfaccia grafica, più intuitiva ma più pesante sotto l'aspetto delle prestazioni, che da linea di comando, più ostica per i meno esperti ma sicuramente più performante.

In questo PoC viene utilizzata una blockchain locale creata tramite ganache-cli, che in fase di inizializzazione invia dei fondi ad alcuni indirizzi tra cui quello che rappresenta ENEA. Questo account viene utilizzato dal server Flask per deployare i contratti durante la fase di startup del sistema e perciò ne risulta l'owner, per questo motivo può eseguire alcuni metodi del contratto (es. minting) riservati esclusivamente al proprietario. Gli accounts pre-funded sono:

ADDRESS	PRIVATE KEY
0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1 (100 ETH, usato per deploy smart contracts)	4f3edf983ac636a65a842ce7c78d9aa706d3b113bce9c46f30d7d21715b23b1d
0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0(100 ETH)	6cbed15c793ce57650b9877cf6fa156fbef513c4e6134f022a85b1ffdd59b2a1
0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b (100 ETH)	6370fd033278c143179d81c5526140625662b8daa446c22ee2d73db3707e620c
0xE11BA2b4D45Eaed5996Cd0823791E0C93114882d(100 ETH)	646f1ce2fdad0e6deeeb5c7e8e5543bdde65e86029e2fd9fc169899c440a7913
0xd03ea8624C8C5987235048901fB614fDcA89b11	add53f9a7e588d003326d1cbf9e4a43c061aadd9bc

7(100 ETH)	938c843a79e7b4fd2ad743
0x95cED938F7991cd0dFcb48F0a06a40FA1aF46EBC (100 ETH)	395df67f0c2d2d9fe1ad08d1bc8b6627011959b79c53d7dd6a3536a33ab8a4fd
0x3E5e9111Ae8eB78Fe1CC3bb8915d5D461F3Ef9A9(100 ETH)	e485d098507f54e7733a205420dfddbe58db035fa577fc294ebd14db90767a52
0x28a8746e75304c0780E011BEd21C72cD78cd535E(100 ETH)	a453611d9419d0e56f499079478fd72c37b251a94bfde4d19872c44cf65386e3
0xACa94ef8bD5ffEE41947b4585a84BdA5a3d3DA6E(100 ETH)	829e924fdf021ba3dbbc4225edfece9aca04b929d6e75613329ca6f1d31c0bb4
0x1dF62f291b2E969fB0849d99D9Ce41e2F137006e (100 ETH)	b0057716d5917badaf911b193b12b910811c1497b5bada8d7711f758981c3773

Tabella 4. Pre-funded accounts

La blockchain di ganache-cli viene lanciata con il flag **-d** (*deterministic*), perciò ad ogni esecuzione gli account rimangono gli stessi.

Smart Contracts

Il progetto contiene i seguenti smart contracts:

- **Verifier**
  Verifica on-chain la prova creata tramite ZoKrates, viene generato automaticamente a partire dal circuito
  - `verify(proof)` → verifica la prova

- **ENEAToken**
  Contratto ERC20 che gestisce il minting verificato dei token e salva le prove
  - `register(userID,userAddress)` → salva on-chain l'associazione id utente - address utente, il minting verificato può essere eseguito solo per utenti registrati [onlyOwner]
  - `mint(amount,to)` → minting classico che invia <amount> tokens all'indirizzo <to> [onlyOwner]
  - `verifiedMint(proof,date,userID,mt)` → verifica <proof> e se la verifica va a buon fine assegna i tokens indicati dalla prova all'address registrato per <userID>; infine salva <proof> in corrispondenza di <date> [onlyOwner]
  - `getVerifiedMint(userAddress,date)` → recupera prova, amount e timestamp dell'utente salvati in corrispondenza del giorno <date>
  - `addressToUser(userAddress)` → fornisce l'id utente corrispondente a <userAddress>
  - `userAddress(userID)` → fornisce l'address corrispondente a <userID>

- `getVerifiedMintDates(addr)` → recupera le date per cui l'utente corrispondente ad <addr> ha dei mintings verificati

I contratti devono essere deployati rigorosamente nell'ordine 1. Verifier 2. ENEAToken in quanto l'address del contratto Verifier deve essere passato nel costruttore di ENEAToken.

3.5 Backend

Il backend, data la complessità delle operazioni svolte, è l'unico container ad avere realmente una cartella dedicata nel repository (in realtà anche ganache ma contiene solo il Dockerfile) con tutti i files necessari al suo funzionamento. Questo modulo utilizza la nota libreria Python *Flask* e svolge le seguenti funzioni:

- serve due interfacce web, una che rappresenta la dashboard di ENEA e l'altra la pagina riservata agli utenti
- genera le prove tramite ZoKrates (installato nativamente nel container)
- interagisce con la blockchain
- interagisce con il db

La cartella `/backend` ha la seguente struttura:

`/backend`

`/abi` → contiene le ABI dei contratti in uso

`/circuits` → contiene i circuiti ZoKrates compilati

`/proving_keys` → contiene la proving key del circuito (generata con zokrates setup)

`/verification_keys` → contiene la verification key del circuito (generata con zokrates setup)

`/static` → contiene i file statici serviti da Flask

`/templates` → contiene i templates HTML utilizzati da Flask per costruire le pagine web

`Dockerfile` → descrive l'immagine Docker del backend

`calc_tokens.zok` → programma ZoKrates per la generazione della prova

`mimc_hash.zok` → programma ZoKrates per calcolare l'hash MiMC

`requirements.txt` → dipendenze Python richieste

`config.json` → contiene la chiave privata dell'account che rappresenta ENEA

`commands_db.py` → modulo Python che raccoglie tutte le funzionalità legate al database

`commands_eth.py` → modulo Python che raccoglie tutte le funzionalità legate alla blockchain

`commands_zok.py` → modulo Python che raccoglie tutte le funzionalità legate a ZoKrates

`main.py` → programma Python principale

A meno di modifiche al circuito `calc_tokens.zok` i seguenti dati

- verification key
- proving key
- contratto Verifier
- circuiti compilati in `/circuits`

restano costanti, perciò si è deciso di non rigenerarli ogni volta che viene avviato il sistema in modo da non allungare eccessivamente i tempi di inizializzazione.

Dashboard ENEA

Endpoint: /

Descrizione: per agire on-chain utilizza l'account corrispondente alla chiave privata contenuta nel file di configurazione, che rappresenta l'account di ENEA

Azioni:

- cambio parametri conteggio (giorno,utente,mt,sc)
  possibilità di cambiare i parametri utilizzati nell'algoritmo per il calcolo dei tokens, ossia data, id utente, MT e SC
- registrazione utente on-chain
  creazione di un'associazione on-chain tra id utente e address utente
- generazione prova
  creazione di una Zero-Knowledge Proof che certifica il numero di penalità conteggiate relativamente all'utente e al giorno selezionati
- minting classico
  minting tradizionale senza l'impiego di ZKPs
- minting verificato
  minting previa verifica della ZKP, che viene salvata on-chain e può essere letta a posteriori

Dashboard utente

Endpoint: /user

Descrizione: simula una dashboard per gli utenti che permette di collegarsi alla pagina tramite MetaMask per visualizzare il saldo tokens corrente e leggere le prove salvate on-chain

Azioni:

- lettura date per cui l'utente ha dei verified mintings
  visualizzazione delle date per cui sono stati eseguiti dei verified mintings e quindi sono disponibili delle prove
- lettura prova salvata on chain per un determinato giorno
  lettura della prova salvata dallo smart contract in occasione di uno specifico verified minting

ZoKrates è un toolbox che permette di generare ZKPs e di verificarle su rete Ethereum. All'interno del PoC ZoKrates viene installato in modo nativo all'interno del container backend, creato partendo dall'immagine base di Linux Ubuntu 20.04, e viene utilizzato principalmente attraverso il server Flask.

Circuito principale

`/backend/calc_tokens.zok`

INPUT	PRIV/PUB	TYPE	DESCR
readings	private	field[96]	consumi quartorari utente
commitment	public	field	merkle root consumi
sc	public	field	soglia carico massimo
penalties	public	field	numero penalties accumulate per la giornata

Tabella 5. Inputs circuito ZoKrates

Il programma ZoKrates `/backend/calc_tokens.zok` può essere suddiviso in due parti principali. Nella prima parte viene ricostruito il Merkle tree dei consumi dell'utente e viene confrontato il suo root con il commitment (che è un ingresso pubblico), con il fine di verificare i dati utilizzati nel calcolo. Nella seconda parte invece viene effettuato il conteggio vero e proprio per determinare se il numero di penalità che è stato calcolato in precedenza è corretto.

Durante la scrittura di un programma ZoKrates è molto importante valutare con attenzione la tipologia di hash che si intende utilizzare, in quanto alcuni algoritmi sono caratterizzati da un'elevata complessità moltiplicativa che si traduce in un drastico peggioramento delle performance e in un incremento rilevante nelle dimensioni delle prove generate. Per questo motivo per la costruzione del Merkle tree dei consumi si è deciso di utilizzare l'hash MiMC, considerato appunto *ZKP-friendly* in quanto garantisce ottime prestazioni.

Conclusioni

Il prototipo oggetto del presente documento implementa una strategia per ricompensare gli utenti che adottano un comportamento virtuoso dal punto di vista energetico in modo certificato e verificabile, attraverso l'utilizzo di tecnologie innovative quali blockchain, smart contracts e zero-knowledge proofs. In questo scenario per ogni minting verificato gli utenti sono in grado di verificare i dati di consumo utilizzati nell'algoritmo di conteggio dei tokens e di controllare se il calcolo è stato eseguito secondo la procedura prestabilita; questo rappresenta un grande passo avanti per gli utenti rispetto ad un minting standard, in cui non viene offerta sostanzialmente nessuna garanzia sulla correttezza del numero di tokens assegnati.

Abbreviazioni ed acronimi

- ZKP: Zero-Knowledge Proof
- PoC: Proof of Concept

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: Francesco Bruschi, Manuel Tumiati, Vincenzo Rana, Mattia Bianchi, Donatella Sciuto

    year: 2020
  - title: "Acknowledging Value of Personal Information: a Privacy Aware Data Market for Health and Social Research"
    authors: Francesco Bruschi, Vincenzo Rana, Alessio Pagani, Donatella Sciuto
    year: 2020
  - title: "Le applicazioni delle nuove tecnologie: criptovalute, blockchain e smart contract"
    author: Francesco Bruschi
    year: 2020

- *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

- *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: “Tunnelling Trust into the Blockchain: a Merkle Based Proof System

    for Structured Document”
    authors: F. Bruschi, V. Rana, A. Pagani, D. Sciuto
  - title: “Mine with it or sell it: the superhashing power dilemma”
    authors: F. Bruschi, V. Rana, L. Gentile, D. Sciuto