

Elaborato esame di stato 2020/2021

Autore: Toniolo Riccardo

Introduzione

In questo documento presenterò come sono riuscito a costruire da zero una suite di protocolli in Php, utilizzati in una web app, in grado di effettuare transazioni di una criptovaluta fittizia, registrate su una struttura dati di tipo blockchain.

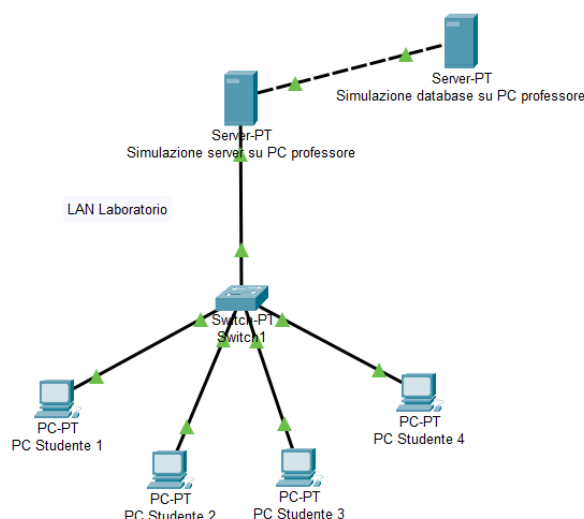
A cosa potrà essere utile?

Potrà essere utilizzata dai professori di Sistemi e Reti per fare capire e notare agli studenti, un utilizzo concreto di: funzioni di hash, chiavi pubbliche e private, RSA e firme, grazie a degli strumenti appositi integrati all'interno della web app.

Infrastruttura di rete

L'applicazione nasce, come già detto in precedenza, per essere utilizzata dai professori durante le ore pratiche di laboratorio per vedere nel concreto alcuni concetti di crittografia, studiati nella parte finale del programma di quinta. Da ciò si può dedurre che l'infrastruttura della rete sarà una **LAN** (Local Area Network) dove l'applicazione verrà messa in "**hosting**", sulla simulazione di un server Apache (ad esempio in XAMPP) all'interno del computer del professore, di modo da permettere un accesso veloce ai vari file dell'applicazione per riuscire ad analizzarli in modo dinamico e approfondito.

La rete dovrebbe avere una forma di questo tipo:



Quali problemi possono verificarsi?

Siccome normalmente i computer Windows non prevedono la ricezione di traffico in entrata, poiché questo è bloccato dal firewall del sistema operativo, per riuscire a risolvere questo problema basterà andare nel pannello di controllo di Windows, entrare nella sezione del firewall e:

- Creare un'**ACE** (Access Control Entry), che permetterà al processo XAMPP di ricevere traffico in entrata;

- Per evitare alcuni problemi che potrebbero verificarsi è meglio creare un'ACE anche per l'uscita di traffico da XAMPP verso la rete;
- Attivare le ACE nella lista di **ACL** (Access Control List) presenti nel firewall.

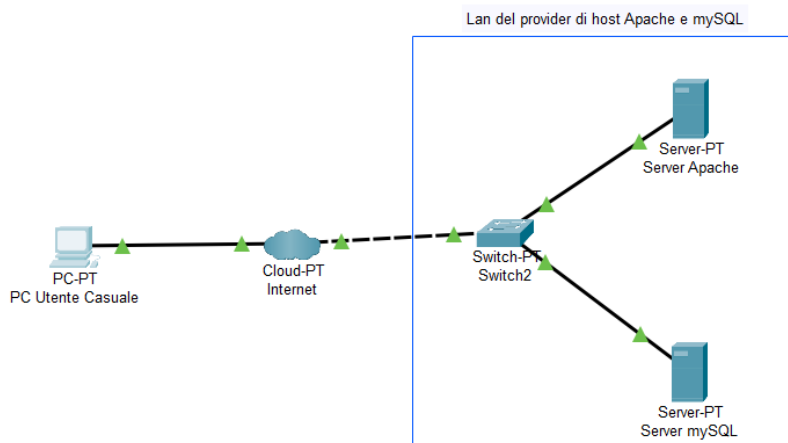
Perché utilizzarlo in una LAN?

Utilizzare il programma mettendolo in hosting in una virtualizzazione di un server all'interno del computer del professore, permetterà al professore stesso di poter apportare modifiche al file contenente la blockchain, di modo riuscire a generare delle reazioni, verificabili all'interno del sito, che permettono di capire principi come la firma digitale. Oltre a questo, è anche facilitata l'analisi dell'applicazione Php stessa.

Dove ho realmente messo in hosting il programma?

Per riuscire a fare provare il programma in via remota, ho hostato il progetto all'interno di un server Apache preso in affitto dal servizio di hosting di Altrivista, successivamente agendo allo stesso modo anche per il server che fa da database.

La struttura di rete dovrebbe essere la seguente (viene previsto l'attraversamento di Internet):



Quali problemi sorgono con l'host in remoto?

Mentre l'host in remoto ci permette di non preoccuparci di avere prestazioni sul pc in grado di sostenere la virtualizzazione di due server, ci risparmia di andare ad aprire ogni volta i servizi, è accessibile ovunque e non richiede la creazione di alcun tipo di ACE sul nostro computer personale, ha comunque una grande carenza.

Il progetto infatti nasce per permettere al professore di andare a modificare il file contenente la blockchain per vedere come il programma stesso si accorge e reagisce a determinati errori. Questo in un server in remoto non è possibile, o comunque viene reso un processo lento e macchinoso.

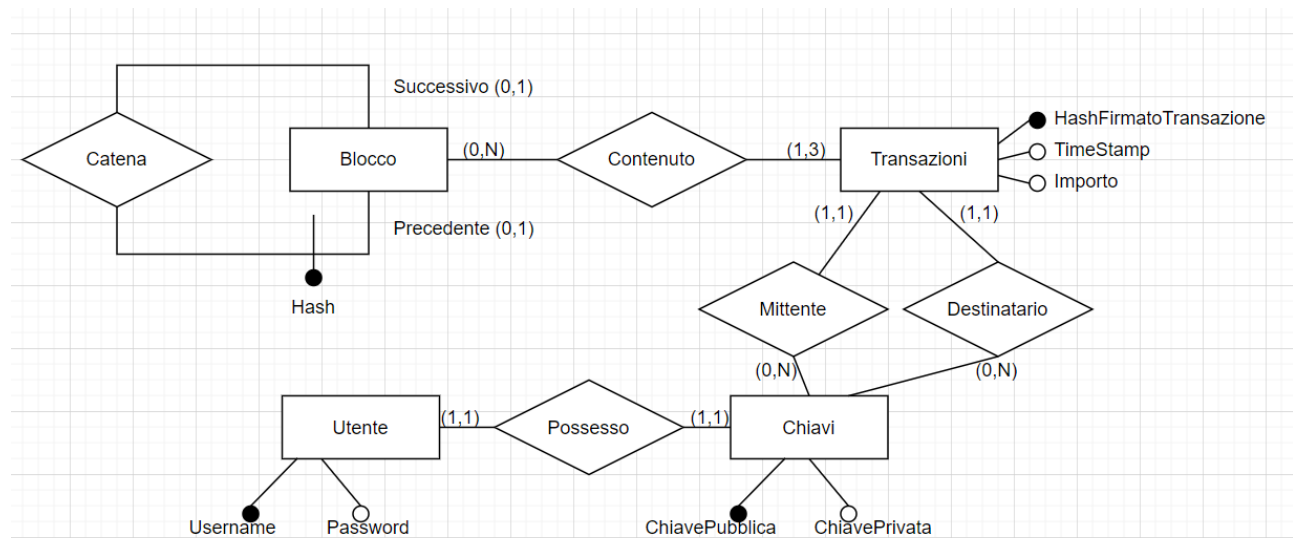
Sviluppo della base di dati

Qual è la complessità del progetto?

Ho dovuto rendere possibile l'interoperabilità tra un sistema di archivio su file JSON e un database strutturato in MySQL.

Schema ER

Lo schema ER della base di dati nel suo complesso è realizzato nel seguente modo:

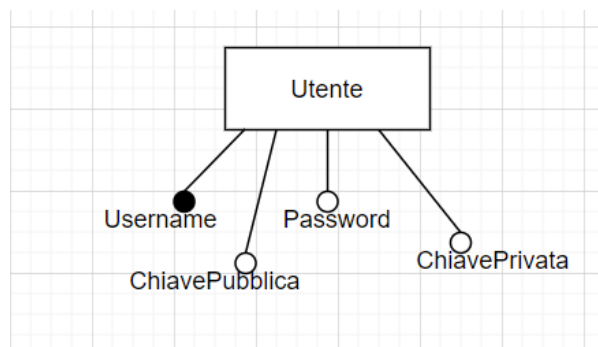


L'ho realizzato nel seguente modo per il principio di astrazione degli schemi ER, poiché in questi non deve trasparire alcuna preferenza di modello logico da utilizzare, andando però a descrivere una realtà di interesse concreta.

Successivamente sono passato alla ristrutturazione, ma per riuscire a farla ho dovuto separare la parte destinata a MySQL da quella JSON (per riuscire a creare degli schemi ER ristrutturati che siano però più simili al modello logico a loro destinato).

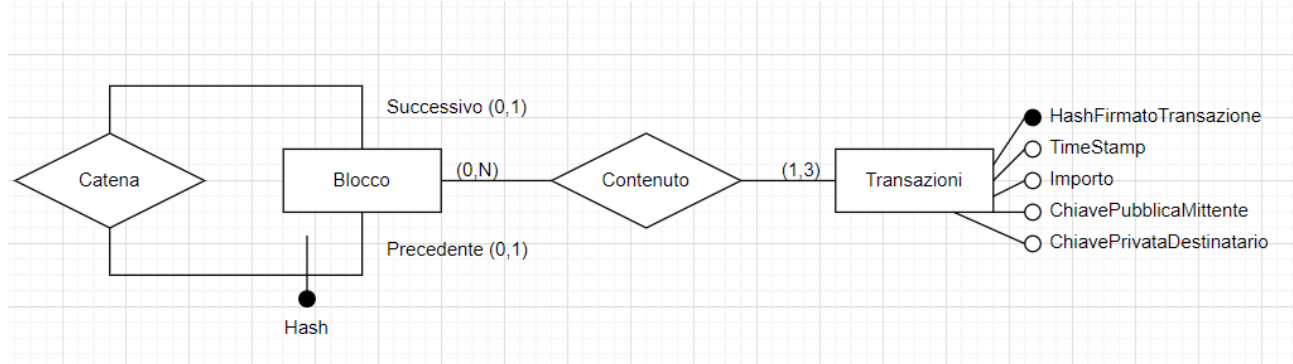
Ristrutturazioni degli schemi ER

Per la parte relativa a MySQL, lo schema ER ristrutturato è questo:



In questo modo tutto quello a cui il database sarà destinato sarà lo storage degli utenti e le loro relative chiavi private e pubbliche (ho accorpato la tabella "Chiavi", con i suoi attributi, alla tabella "Utente" con i relativi attributi), scegliendo come chiave primaria dell'entità lo Username.

Per quanto riguarda la parte di archivio in JSON, lo schema ER ristrutturato è il seguente:



In questo caso invece ho trasformato quelle che erano le relazioni tra la tabella “Chiavi” e la tabella “Transazioni” in attributi di quest’ultima, poiché queste due tabelle non possono più essere collegate assieme, in quanto appartenenti a sistemi di archivio diversi.

La relazione “Mittente” è diventata l’attributo ChiavePubblicaMittente, mentre la relazione “Destinatario” è diventata l’attributo ChiavePubblicaDestinatario.

Così facendo, andrò sicuramente a creare della ridondanza sul sistema JSON, ma riuscirò sia a separare con successo i due metodi di archivio, sia a renderne possibile l’interoperabilità.

Traduzioni degli schemi ER ristrutturati

La traduzione relativa alla parte **relazionale** del database è la seguente:

Utente (Username, Password, ChiavePubblica ,ChiavePrivata)

Le istruzioni per la creazione di questa tabella in MySQL sono le seguenti:

```
CREATE TABLE UTENTI(  
    IdUtente char(64) PRIMARY KEY NOT NULL,  
    Password char(64) NOT NULL,  
    ChiavePrivata TEXT NOT NULL,  
    ChiavePubblica TEXT NOT NULL  
)  
ENGINE=INNODB;
```

Ho utilizzato il tipo TEXT su “ChiavePrivata” e su “ChiavePubblica” in quanto la loro lunghezza va a sfiorare quelli che sono i limiti di lunghezza possibili sia per il tipo “char()” che per il tipo “varchar()”.

Per quanto riguarda la traduzione in un modello JSON dello schema ER corrispondente, questo è il risultato:

```
[  
  "Blocco": [  
    "Transazioni": [  
      {  
        "Mittente":  
        "Destinatario":  
        "Importo":  
        "Timestamp":  
        "Hash firmato":
```

```
    }  
  ],  
  "hashPrecedente":  
]  
]
```

Per chi non fosse familiare con i sistemi di archivio JSON, quanto appena scritto viene interpretato in questo modo:

*La blockchain è formata da blocchi. Ogni blocco è formato da un campo array "Transazioni" e da un campo "hashPrecedente". Ogni transazione è formata da un campo "Mittente", un campo "Destinatario", un campo "Importo", un campo "Timestamp", un campo "Hash firmato" (**questo per lo meno in linea generale, poiché si può verificare un'eccezione dove non compare il campo "Hash firmato" all'interno di alcune transazioni speciali**).*

Perché creare un sistema di archivio basato su un file JSON e non su un server che fornisce un servizio database?

Nei sistemi decentralizzati basati su blockchain, la struttura dati delle transazioni è scritta all'interno di un file testuale poiché, successivamente alla creazione di ogni blocco, il miner (ovvero colui che crea il blocco) ha la necessità di spedire in broadcast a tutti i nodi della rete l'intero libro mastro delle transazioni. Non c'è modo più veloce ed efficace che trasferire un file testuale. Non dovrebbe essere utilizzato un server per registrare le varie transazioni poiché, utilizzando un database basato su un server o più, si andrebbe a centralizzare l'intera rete sui server di chi li possiede (e quindi il tutto non apparterebbe più all'intera rete, ma a uno o pochi individui).

Come trattato nella parte di "Infrastruttura di rete" la mia rete è centralizzata, quindi i motivi per cui ho scelto di utilizzare una struttura dati basata sul file sono:

- Il voler sperimentare una nuova tecnologia di archivio dati, non studiata a scuola;
- Il voler essere quanto più conforme ai metodi tradizionali.

Che vantaggi ha l'uso di un file JSON?

Un file JSON ha il vantaggio di avere delle integrazioni speciali con PHP che permettono a questo linguaggio di leggere il file come fosse un array associativo, riuscendo a trovare e a trattare i dati in modo molto veloce.

Un altro vantaggio è il fatto che la sua struttura è variabile, e quindi può essere organizzata in modo da non creare ridondanza di NULL nel caso in cui non si abbia un valore da inserire in un determinato campo, come ad esempio vedremo dopo, nella parte di "Sviluppo della blockchain". Tutto ciò che semplicemente va a fare è non creare il campo per quella determinata eccezione.

Che svantaggi ha l'uso di un file JSON?

Lo svantaggio principale di usare un file di questo tipo è il fatto che, ogni qual volta che si ha la necessità di modificare qualcosa all'interno del file, o semplicemente anche leggerlo, esso deve essere caricato per intero in memoria primaria. Questo potrebbe creare problemi nel momento in cui il file raggiunga dimensioni molto grandi e il computer adibito ad aprirlo abbia una carenza nella memoria RAM.

Approfondimento sulle blockchain

Cos'è una blockchain?

Una blockchain è una struttura di archivio dati basata su dei blocchi, dove ognuno di questi è legato al precedente avendo all'interno di sé stessi un campo che identifica il blocco precedente in modo univoco attraverso, tipicamente, una funzione di hash applicata all'intero blocco precedente.

In questo modo, se si va a modificare il contenuto di un blocco, si andrà a modificare quello che sarebbe l'hash del blocco corrente, finendo per dare origine ad una blockchain contraffatta, poiché grazie all'effetto cascata delle funzioni di hash, il blocco successivo al blocco contraffatto avrà come hash del blocco precedente, un contenuto non più valido e così via con i blocchi a venire.

Questo libro mastro di informazioni è posseduto da tutti i nodi della rete, e per essere modificato si deve avere il consenso di almeno il 51% della rete stessa (questo nei sistemi decentralizzati).

Visto che un blocco possiede un campo che contiene l'hash dell'intero blocco precedente, come funziona il primo blocco?

Il primo blocco, anche detto blocco genesi, è trattato in modo speciale dalle funzioni del programma, in quanto è l'unico blocco avente come valore di "hashPrecedente" il valore "0". Su di esso, tuttavia, i protocolli per l'integrità dei blocchi non variano.

Come fa la mia blockchain a gestire il consenso?

Nelle blockchain decentralizzate il consenso viene gestito dai nodi della rete che sono adibiti alla creazione di blocchi tramite complessi algoritmi matematici, i quali vengono chiamati anche "Miner".

Tuttavia, nel caso della mia blockchain, essendo invece centralizzata, l'unico creatore di blocchi è il server centrale che esegue gli script di Php. Non essendoci quindi una competizione alla creazione dei blocchi con altri miner, tutti si fidano del server centrale, in pratica eliminando il problema del consenso.

Quali sono i casi d'uso di una blockchain?

I casi d'uso sono veramente tanti, ma tutto nacque nel 2008 quando Satoshi Nakamoto rilasciò pubblicamente dei documenti che descrivevano il protocollo per effettuare transazioni in modo decentralizzato di Bitcoin. Infatti, le blockchain nascono per effettuare transazioni di valute in grado però di non passare per un ente centralizzato come ad esempio una banca.

Al giorno d'oggi le blockchain sono utilizzate per fornire diversi tipi di *servizi*:

- servizi di finanza decentralizzata, come ad esempio il fornire o richiedere prestiti;
- servizi di logistica (BMW e Microsoft già usano queste tecnologie nelle loro aziende);
- servizi per l'anticontraffazione, o meglio, servizi per la certificazione di originalità di beni (anche realmente esistenti).

Questo perché il contenuto di una catena di blocchi può essere qualsiasi cosa si voglia, a seconda dell'esigenza che si ha.

Cos'è il contenuto dei blocchi nel caso del mio progetto?

Il contenuto dei blocchi in questo caso sono transazioni aventi i seguenti campi:

- **Mittente**: è la chiave pubblica del mittente, ovvero colui che volendo mandare una transazione, una volta assemblata, avrà firmato con la propria chiave privata l'hash della transazione stessa;

- **Destinatario:** è la chiave pubblica del destinatario, usata semplicemente come identificatore di quest'ultimo;
- **Importo:** importo della transazione;
- **Timestamp:** numero di secondi dal 1° gennaio 1970 al momento della transazione;
- **Hash firmato:** campo firmato con la chiave privata del mittente. Serve per garantire autenticità e verifica dell'integrità della transazione stessa.

Come viene gestita nel mio programma una blockchain contraffatta?

Viene lasciata come tale la blockchain in sé, ma accadono due eventi:

- Tutte le transazioni modificate, non verranno calcolate dai calcolatori del saldo dei vari conti;
- Nella pagina di scanner della blockchain, utilizzata per fare vedere come il programma individua transazioni contraffatte, verranno mostrati in rosso i blocchi manomessi e in arancione le transazioni modificate.

Nel caso in cui si provi a fare un copia e incolla di una transazione precedente, il programma è in grado di accorgersene e di conseguenza non la conterà quando andrà a calcolare il saldo degli utenti ad essa collegati.

Sviluppo del progetto

Come dovrebbe funzionare l'applicazione?

Un utente, una volta registrato e acceduto al suo account potrà andare a generarsi, sottrarsi o trasferire EDUCoin.

Le tre operazioni descritte sopra implicano la creazione di una transazione all'interno del file di blockchain. Per creare una transazione però, nel caso in cui il file contenente la blockchain non esistesse, bisognerebbe:

- Crearlo;
- Creare al suo interno il blocco genesis;
- Solo infine registrare la transazione all'interno del blocco.

Ogni blocco contiene al massimo tre transazioni (questo l'ho impostato io dentro la funzione adibita alla creazione dei blocchi, per permettere di creare quanti più blocchi possibili da poter analizzare, nel minor tempo possibile). Quindi nel caso in cui si tenti di immettere una transazione nell'ultimo blocco disponibile ma esso contenga già tre transazioni, bisognerà creare un ulteriore blocco con, come valore di hashPrecedente, l'hash dell'intero blocco di prima, per poi inserire la nuova transazione come prima transazione del nuovo blocco.

Cos'è un wallet e da cosa è costituito?

Un wallet è l'insieme della chiave privata e della chiave pubblica, appartenenti a un utente. Queste serviranno rispettivamente per firmare l'hash della transazione e per verificare la transazione stessa.

C'è anche da dire che la chiave pubblica, quando usata nelle transazioni, viene anche utilizzata nel campo destinatario per identificare in modo univoco il destinatario a cui spetta l'ammontare dell'importo.

Queste chiavi vengono generate e associate automaticamente ad un utente da una funzione del programma, utilizzata quando un utente viene registrato con successo.

Ho utilizzato la libreria PHPSECLIB per generare le varie chiavi, sfruttando il sistema RSA.

In questo caso, la creazione della chiave privata da PHPSECLIB avviene tramite la generazione di un numero pseudocasuale definito da dell'entropia, poi codificato in forma di chiave alfanumerica. La chiave pubblica invece, viene ricavata dalla prima chiave, inserendo quel numero pseudocasualmente generato in una funzione di curva ellittica, il cui risultato sarà un numero che, una volta codificato in modalità alfanumerica, rappresenterà la chiave pubblica stessa.

Non ho usato la libreria di default di Php per utilizzare l'RSA in quanto, oltre a darmi parecchi problemi con l'esecuzione dei metodi, risulta più difficile e lenta sia da capire che da applicare.

Che cosa cambia tra la registrazione di una transazione in un database, rispetto ad una blockchain?

I sistemi decentralizzati basati su blockchain posseggono, come i DBMS le seguenti caratteristiche:

- Sono in grado di elaborare grandi quantità di dati;
- I dati sono condivisi, con tutti i nodi della rete;
- I dati sono persistenti in un file che fa da libro mastro per tutte le transazioni, registrato nelle memorie di tutti i nodi della rete;
- In caso di eventuali disastri, essendo che il file è condiviso da tutti i nodi della rete, non si teme che i dati vengano persi, perché è presente una copia di backup in ogni nodo della rete;
- I sistemi di ultima generazione sono efficienti poiché ottimizzano l'uso delle risorse a disposizione della rete;
- Sono efficaci in quanto utilizzano sistemi evoluti, per raggiungere la diffusione, convalida e conferma delle transazioni in poco tempo.

Oltre a quanto elencato però, utilizzano i sistemi di crittografia per rendere le singole transazioni verificate a livello di integrità ed autenticate, senza mettere di mezzo un'entità centrale che le analizza, attuando il principio della firma digitale (Utilizzando la chiave privata del mittente per firmare l'hash della transazione da lui eseguita). In questo modo si può dimostrare che solo se la chiave del mittente è quella giusta, l'hash della transazione verrà verificato.

In aggiunta a questo, gli utenti sono riconducibili solamente a numeri, mantenendo anonima la propria identità.

La funzione per la verifica di una transazione è la seguente:

```
function verifyTransaction($transazione){
    if($transazione["Mittente"]=="NETWORK" or $transazione["Destinatario"]=="NETWORK"){
        return true;
    }else{
        if(hash("sha256",serialize(array($transazione["Mittente"],$transazione["Destinatario"],$transazione["Importo"],$transazione["Timestamp"])))
        )===decryptRSAKeyGiven($transazione["Mittente"],utf8_decode($transazione["Hash firmato"]))) {
            return true;
        }else{
            return false;
        }
    }
}
```

Ammenoché la transazione non provenga o sia destinata alla NETWORK (Quindi si parla di generazione o eliminazione di moneta), si verifica che l'hash della transazione corrisponda con l'hash ricavato dalla decifratura con la chiave pubblica del mittente, del campo di transazione "Hash firmato". Solo in tal caso ritorna "true", altrimenti la transazione si dice non valida.

Come si aggiunge moneta al sistema, di modo da permettere la creazione di transazioni?

Tramite la pagina “Genera” del progetto, si potrà chiedere al server la creazione di una transazione destinata a noi (quindi con la nostra chiave pubblica come destinatario). Siccome però, il server non possiede chiavi pubbliche o private, scriverà nel campo mittente il valore “NETWORK”, e non includerà all’interno della transazione il campo Hash Firmato, visto che appunto non possiede nessuna chiave con cui firmare.

Nel mondo reale, l’immissione di moneta avviene in modi totalmente diversi e viene regolata da complessi algoritmi che richiedono l’impiego di macchine capaci di alte prestazioni computazionali. Io ho deciso di strutturare il progetto in questo modo per i seguenti motivi:

- Rendendo la generazione di moneta illimitata, rendo impossibile il riutilizzo dell’intero programma per scopi di lucro, visto appunto che il progetto è destinato all’educazione;
- Rendo molto più veloci e leggeri i processi per l’esecuzione delle funzioni Php sul server, visto che di solito la generazione di nuova moneta è originata dall’attività di mining, ovvero un’attività molto onerosa e lenta in termini di prestazioni e tempi.

Come si comporta il programma nella creazione della prima transazione all’interno della blockchain, quando ancora il file non esiste?

Per rispondere a questa domanda bisogna capire quali problemi potrebbero presentarsi durante questo passaggio:

- Il file non esiste, quindi lo si va a generare, per poi creare il primo blocco ed infine inserire la prima transazione. **SOLUZIONE:** bisogna creare un metodo per la verifica o meno dell’esistenza dei file;
- Due processi verificano che il file non esiste, quindi il primo lo crea e registra la transazione e immediatamente dopo il secondo processo lo ricrea, per poter registrare la propria transazione (andando a perdere la transazione del primo). **SOLUZIONE:** bisogna far sì che il file nel caso erraneo in cui si tentasse di crearlo (quando in realtà già esiste), venga solamente aperto come se fosse in lettura.

Per risolvere i seguenti problemi basta applicare la seguente funzione, all’inizio di qualsiasi altra funzione che prevede la lettura o scrittura della blockchain:

```
function verifyFile(){
    if(!file_exists("blockchain.json)){
        $blockchain = fopen("blockchain.json","c+");
        fclose($blockchain);
        return false;
    }else{
        return true;
    }
}
```

Tramite la seguente funzione, se il file non esiste, verrà creato attraverso il metodo “c+”, il quale in questo caso agirebbe come il metodo di apertura “w” (Andando a creare il file nel caso in cui non esista).

Nel malaugurato caso in cui il file esistesse già, il metodo “c+” prevede di non ricrearlo da capo, e non prevede nemmeno di cancellare quanto già scritto, semplicemente si limita ad aprire il file o per aggiungere testo in coda, oppure per effettuare operazioni di lettura.

Nel caso della multiutenza può accadere che due utenti tentino contemporaneamente di creare un nuovo blocco per aggiungere una nuova transazione, come gestisco questo problema?

Ho creato un file, il cui unico scopo è quello di regolare la mutua esclusione per quanto riguarda l'accesso in scrittura al file contenente la blockchain. Nel caso qualcuno abbia il bisogno di aggiungere una nuova transazione, dovrà appropriarsi del lock utilizzando il metodo Php flock() sul file di lock, e solo in quel momento dovrà verificare se aggiungere un nuovo blocco oppure, se possibile, aggiungere la transazione direttamente in un blocco già esistente, con meno di tre transazioni al suo interno.

A questo punto potrebbe sorgere una domanda: ma allora come mai non hai messo il lock direttamente sul file contenente la blockchain?

Alcune funzioni, tipo quelle per il calcolo del saldo, hanno bisogno di accedere in qualsiasi caso alla blockchain, e il mancato accesso a questa determinerebbe un errore sulle pagine.

Mettere il lock direttamente sul file di blockchain andrebbe a bloccare la lettura di qualsiasi altro processo, nel caso in cui qualcun altro lo stia leggendo o modificando, generando errori ed eccezioni di vario tipo.

Quindi a costo di dover avere delle potenziali *letture sporche*, che non arrecano danno a nessuno (poiché non si basano sulla lettura per la modifica di valori preesistenti), ho preferito fare in modo che il file sia sempre disponibile in lettura, limitando l'accesso solo in scrittura.

Il metodo è il seguente, applicato sul metodo di creazione di una transazione:

```
function addTransaction(string $destinatario,$importo){
    verifyFile();
    $lock = fopen("Lock.txt","c+");
    while(flock($lock,LOCK_EX)===FALSE){
    }
    if(VerifyTransactionsInABlock()){
        createNewTransaction($destinatario,$importo);
    }else{
        createNewBlock();
        createNewTransaction($destinatario,$importo);
    }
    fclose($lock);
}
```

Glossario

ACE (Access Control Entry): è una singola regola che esprime le regole di accesso ad una risorsa.

Le ACE hanno sempre:

- Un obbiettivo: cosa farne del pacchetto;
- Un'interfaccia: specifica su quale interfaccia e in quale verso vige la regola che si intende applicare;
- Delle specifiche: si specificano le caratteristiche del pacchetto.

ACL (Access Control List): è un insieme di ACE, che usualmente fanno riferimento ad uno stesso argomento. Ad esempio, l'insieme di tutte le ACE che separano la rete trust dalla rete DMZ comporranno l'ACL per la separazione delle due reti.

Firewall: è un processo software che controlla il traffico in uscita e filtra il traffico in entrata. Nel caso del firewall di windows si tratta di un "Personal firewall", ma nel caso in cui si stia parlando di un firewall che ha accesso ad una rete pubblica si parla di "Firewall perimetrale".

XAMPP: programma che permette la virtualizzazione di server per l'hosting di diversi servizi, tra cui un server Apache per l'esecuzione di codice PHP e un server MySQL per la costruzione e utilizzo di database.

Timestamp: numero di secondi dal primo gennaio 1970 fino ad oggi.

Database: archivio di dati che nel caso in cui utilizzi il livello logico di tipo relazionale, è un insieme di tabelle correlate fra loro.

MySQL: dialetto del linguaggio originale SQL, utilizzato per creare strutture dati di tipo relazionale.

JSON: tipo di archiviazione dati, registrata all'interno di un file di tipo JSON. Nasce per l'utilizzo dell' http come protocollo di trasporto, per trasportare appunto informazioni non di tipo HTML.

Funzione di hash: funzione tale per cui dato un input, si ottiene un output che lo identifica in modo univoco. Questo output deve però rispettare delle regole:

- Data la modifica di un singolo carattere dell'input, deve avvenire un effetto cascata tale per cui l'output ne risulta completamente modificato;
- La funzione deve essere facile da calcolare;
- La funzione deve generare un output tale per cui, sia praticamente impossibile ricavarne l'input originale;
- Dato l'output di una funzione di hash su un input P, non deve esistere la possibilità che un input P' generi lo stesso output.

Nell'eventualità in cui, si verificasse l'evento descritto nell'ultimo caso, l'algoritmo si può definire violato in quanto sono state generate delle collisioni.

RSA: metodo di generazione di una chiave privata e di estrapolazione da questa di una chiave pubblica. Principalmente usato per la generazione di chiavi asimmetriche, per usarle per firme asimmetriche oppure per cifrare e decifrare in modalità asimmetrica.

SCHEMA ER: schema utilizzato per rappresentare una realtà di interesse in modo astratto, ovvero non andandola ad adattare ad uno specifico modello logico in anticipo.

Gli schemi ER posseggono tre entità fondamentali:

- Il rettangolo: rappresenta le entità;
- Il rombo: rappresenta la relazione tra entità;
- Il cerchio: rappresenta l'attributo di un'entità.

Mutua esclusione: caso in cui due processi hanno la necessità di accedere ad una risorsa comune, ma si vuole fare sì che si possa accedere solo uno alla volta e che colui che accede abbia l'uso esclusivo della risorsa. Questo per evitare anomalie come l'aggiornamento perduto o il dirty reading.

Lock di una risorsa: metodo usato per garantire la mutua esclusione, dove il primo processo che riesce ad accedere alla risorsa, ne acquisirà l'accesso e l'uso esclusivo, e non la renderà libera fino alla fine delle proprie istruzioni da eseguire su di essa, dichiarando in fine il rilascio della risorsa. A quel punto reclamerà il lock il primo thread che riuscirà ad accedervi e a vedere la risorsa libera.

Lettura sporca o Dirty reading: quando un valore viene letto poco prima che esso venga modificato. Quindi non si legge il valore reale, ma un valore non più giusto.

Codice significativo (ritrovabile nella pagina funzioni.php)

Verifica dell'esistenza del file e in caso negativo, generazione di questo:

```
function verifyFile(){
    if(!file_exists("blockchain.json")){
        $blockchain = fopen("blockchain.json","c+");
        fclose($blockchain);
        return false;
    }else{
        return true;
    }
}
```

Generazione coppia di chiavi (privata e pubblica) da utilizzare per la creazione di un wallet:

```
function createNewKeyPair(){
    //Genera private keys da 492 caratteri
    //Genera public keys da 181 caratteri
    $rsa = new Crypt_RSA();
    $rsa->setHash('sha256');

    $keys = $rsa->createKey(512);

    $keyPrivate = $keys["privatekey"];

    $keyPublic = $keys["publickey"];
    return [$keyPrivate,$keyPublic];
}
```

Cifratura di una stringa tramite chiave privata:

```
function encryptRSA($plainText){
    $rsa = new Crypt_RSA();
    $rsa->loadKey($_SESSION["privkey"]);
    $rsa->setEncryptionMode(CRYPT_RSA_ENCRYPTION_PKCS1);
    $chiphertext = $rsa->encrypt($plainText);
    return $chiphertext;
}
```

Decifrazione di una stringa criptata in precedenza da una chiave privata, attraverso l'uso di una chiave pubblica (generata dalla chiave privata corrispondente):

```
function decryptRSA($chiphertext){
    $rsa = new Crypt_RSA();
    $rsa->loadKey($_SESSION["pubkey"]);
    $rsa->setEncryptionMode(CRYPT_RSA_ENCRYPTION_PKCS1);
    $plainText = $rsa->decrypt($chiphertext);
    return $plainText;
}
```

Creazione di un blocco di qualsiasi tipo (sia geni che non geni):

```
function createNewBlock(){
    $blockchainArray = file_get_contents("blockchain.json");
    $blockchainArray = json_decode($blockchainArray,TRUE);
    if(filesize("blockchain.json")){
        $blocco = array("Transazioni"=>array(),"hashPrecedente"=>hash("sha256",serialize($blockchainArray[count($blockchainArray)-1])));
        $blockchainErased = fopen("blockchain.json","w");
        $blockchainArray[]=$blocco;
        fwrite($blockchainErased,json_encode($blockchainArray));
        fclose($blockchainErased);
    }else{
        $bloccoGenesi = array(array("Transazioni"=>array(),"hashPrecedente"=>0));
        $blockchainErased = fopen("blockchain.json","w");
        fwrite($blockchainErased,json_encode($bloccoGenesi));
        fclose($blockchainErased);
    }
}
```

Funzione per la creazione di una transazione generale:

```
function createNewTransaction($destinatario,$importo){
    $timestamp = time();
    $transazione = array("Mittente"=>$_SESSION["pubkey"],"Destinatario"=>$destinatario,"Importo"=>$importo,"Timestamp"=>$timestamp,
    "Hash firmato"=>utf8_encode(encryptRSA(hash("sha256",serialize(array($_SESSION["pubkey"],$destinatario,$importo,$timestamp))))));
    $blockchainArray = file_get_contents("blockchain.json");
    $blockchainArray = json_decode($blockchainArray,TRUE);
    $blockchainArray[count($blockchainArray)-1]["Transazioni"][] = $transazione;
    $blockchainErased = fopen("blockchain.json","w");
    fwrite($blockchainErased,json_encode($blockchainArray));
    fclose($blockchainErased);
}
```

Aggiunta di una transazione da un determinato mittente a un determinato destinatario:

```
function addTransaction(string $destinatario,$importo){
    verifyFile();
    $lock = fopen("Lock.txt","c+");
    while(flock($lock,LOCK_EX)===FALSE){
    }
    if(VerifyTransactionsInABlock()){
        createNewTransaction($destinatario,$importo);
    }else{
        createNewBlock();
        createNewTransaction($destinatario,$importo);
    }
    fclose($lock);
}
```

Creazione di nuova moneta, creando una nuova transazione avente come mittente "NETWORK":

```
function createNewMoney($importo){
    $timestamp = time();
    $transazione = array("Mittente"=>"NETWORK","Destinatario"=>$_SESSION["pubkey"],"Importo"=>$importo,"Timestamp"=>$timestamp);
    $blockchainArray = file_get_contents("blockchain.json");
    $blockchainArray = json_decode($blockchainArray,TRUE);
    $blockchainArray[count($blockchainArray)-1]["Transazioni"][] = $transazione;
    $blockchainErased = fopen("blockchain.json","w");
    fwrite($blockchainErased,json_encode($blockchainArray));
    fclose($blockchainErased);
}
```

Cancellazione di moneta esistente, creando una nuova transazione avente come destinatario “NETWORK”:

```
function eraseMoney($quantitaDaSottrarre){
    $timestamp = time();
    $transazione = array("Mittente"=>$_SESSION["pubkey"],"Destinatario"=>"NETWORK","Importo"=>$quantitaDaSottrarre,"Timestamp"=>$timestamp);
    $blockchainArray = file_get_contents("blockchain.json");
    $blockchainArray = json_decode($blockchainArray,TRUE);
    $blockchainArray[count($blockchainArray)-1]["Transazioni"][] = $transazione;
    $blockchainErased = fopen("blockchain.json","w");
    fwrite($blockchainErased,json_encode($blockchainArray));
    fclose($blockchainErased);
}
```

Verifica della validità della transazione, confrontando l’hash (del campo di transazione “Hash firmato”) decriptato con la chiave pubblica trovata nel campo mittente, con l’hash ricavato dall’applicazione dell’algoritmo SHA-256 sulla transazione stessa:

```
function verifyTransaction($transazione){
    if($transazione["Mittente"]=="NETWORK" or $transazione["Destinatario"]=="NETWORK"){
        return true;
    }else{
        if(hash("sha256",serialize(array($transazione["Mittente"],$transazione["Destinatario"],$transazione["Importo"],$transazione["Timestamp"])))
        )===decryptRSAKeyGiven($transazione["Mittente"],utf8_decode($transazione["Hash firmato"]))) {
            return true;
        }else{
            return false;
        }
    }
}
```

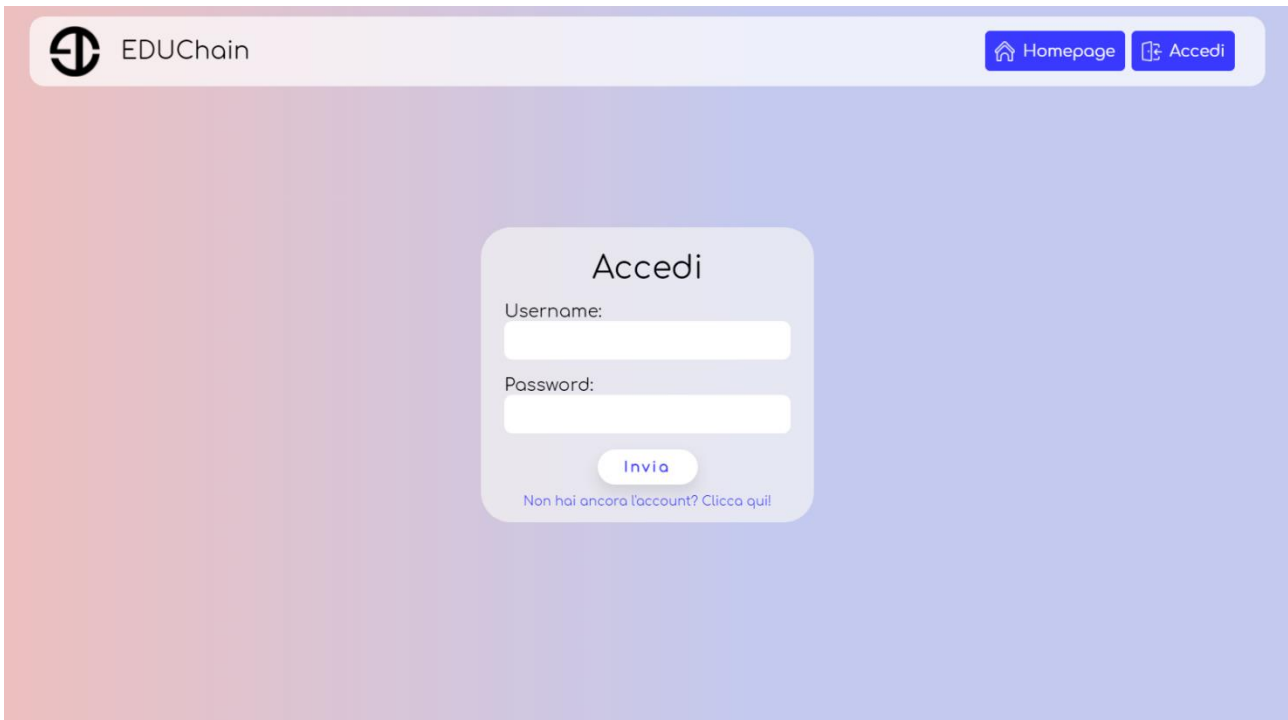
Verifica della validità dell’intera blockchain, confrontando l’hash dei singoli blocchi con l’hash dei blocchi successivi (nel caso dell’individuazione di un blocco contraffatto, che genererebbe un errore a catena con i blocchi seguenti, viene registrato il suo numero all’interno della variabile di sessione \$_SESSION["corruptedBlock"][]), restituendo false nel caso in cui si individuasse un blocco contraffatto:

```
function verifyBlockchain(){
    unset($_SESSION["corruptedBlock"]);
    $isNotCorrupted = true;
    if(file_exists("../blockchain.json")){
        if(filesize("../blockchain.json")){
            $blockchainArray = file_get_contents("../blockchain.json");
            $blockchainArray = json_decode($blockchainArray,TRUE);
            $counter=0;

            foreach ($blockchainArray as $numBlocco => $blocco) {
                if($counter==0){
                    foreach ($blocco["Transazioni"] as $index => $transazione) {
                        if(!verifyTransaction($transazione)){
                            $_SESSION["corruptedBlock"][] = $counter+1;
                            $isNotCorrupted = false;
                        }
                    }
                }else{
                    if($blocco["hashPrecedente"]!==hash("sha256",serialize($blockchainArray[$counter-1]))){
                        $isNotCorrupted = false;
                        $_SESSION["corruptedBlock"][] = $counter;
                    }
                    foreach ($blocco["Transazioni"] as $index => $transazione) {
                        if(!verifyTransaction($transazione)){
                            $_SESSION["corruptedBlock"][] = $counter+1;
                            $isNotCorrupted = false;
                        }
                    }
                }
                $counter++;
            }
        }else{
            $isNotCorrupted = true;
        }
    }else{
        $isNotCorrupted = true;
    }
    return $isNotCorrupted;
}
```

Pagine del programma

Pagina di login:



EDUChain

Homepage Accedi

Accedi

Username:

Password:

Invia

[Non hai ancora l'account? Clicca qui!](#)

Pagina del saldo:



EDUChain

Saldo Invia Ricevi Genera Ritira Blockchain Logout

Il tuo saldo è di:

20ED

Aggiorna

Transazioni:

Sent: 20ED

From: MFwwDQYJKoZIhvcNAQEBBQAI GUarBE65orjczYDRYS0QnsPUS: ---


To: MFwwDQYJKoZIhvcNAQEBBQAI GUarBE65orjczYDRYS0QnsPUS: ---

Received: 40ED

From: NETWORK

To: MFwwDQYJKoZIhvcNAQEBBQAI GUarBE65orjczYDRYS0QnsPUS: ---

Pagina di invio:

 EDUChain

SaldoInviaRiceviGeneraRitiraBlockchainLogout

Seleziona il tipo di ricerca nome:


Scrivi nomeLista nomi

Scrivi a chi vuoi mandare monete EDU:

Quantità:

Invia

Pagina “ricevi”, usata per le informazioni riguardanti l’account:

 EDUChain

SaldoInviaRiceviGeneraRitiraBlockchainLogout

Informazioni personali


Il tuo nome utente:

Riccardo

La tua chiave pubblica:

MFwwDQYJKoZIhvcGUarBE65orjczYDR

Pagina per generazione di monete:

 EDUChain


SaldoInviaRiceviGeneraRitiraBlockchainLogout

Genera monete EDU

Inserisci il numero di monete che desideri ricevere:

Genera

Pagina per ritirare le monete dalla rete:

 EDUChain

SaldoInviaRiceviGeneraRitiraBlockchainLogout

Ritira monete EDU

Inserisci il numero di monete che desideri ritirare dal tuo conto:

Ritira

Pagina per la visualizzazione e analisi della blockchain:

The screenshot shows the EDUChain Blockchain Scanner interface. At the top, there is a navigation bar with the EDUChain logo and several buttons: Saldo, Invia, Ricevi, Genera, Ritiro, Blockchain, and Logout. Below the navigation bar, the status is displayed as "Blockchain status: OK" with a green checkmark. To the right, it shows "Numero di blocchi: 1" and "Numero di transazioni: 2". The main section is titled "Blockchain scanner" and features a "Blocco #1" card. This card contains details for "Transazione #1": Mittente: NETWORK, Destinatario: MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJB, GUarBE65orjczYDRYS0QnsPUS+7wjV9Hh4ej, Importo: 40, and Data: 05/25/2021. Below this, the start of "Transazione #2" is visible.

Visualizzazione della pagina di blockchain nel caso in cui venga manomessa una transazione, andando a compromettere l'intero registro:

The screenshot shows the EDUChain Blockchain Scanner interface with a corrupted status. The navigation bar is the same as in the previous image. The status is now "Blockchain status: Corrupted" with a red warning icon. It indicates "First corrupted block: #1". The "Numero di blocchi" remains at 1, but the "Numero di transazioni" is still 2. The "Blockchain scanner" section shows "Transazione #2" highlighted with a red border, indicating it is the corrupted transaction. The details for "Transazione #2" are: Mittente: MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJB, GUarBE65orjczYDRYS0QnsPUS+7wjV9Hh4ej, Importo: 21, and Data: 05/25/2021. The "Destinatario" field is partially visible but mostly obscured by the red border.