



Università degli Studi di Milano Bicocca

Dipartimento di Informatica, Sistemistica e Comunicazione

Corso di Laurea Magistrale in Informatica

ANALISI E IMPLEMENTAZIONE DI PROTOCOLLI DI VOTO ELETTRONICO SU BLOCKCHAIN CON DIMOSTRAZIONI ZERO-KNOWLEDGE

Relatore: Prof. Alberto Leporati

Co-relatore: Dott. Raffaele Nicodemo

Tesi di Laurea Magistrale di:

Luca Virgilio

Matricola 794866

Anno Accademico 2019-2020

“It is enough that the people know there was an election. The people who cast the votes decide nothing. The people who count the votes decide everything.”

JOSEPH STALIN

Ringraziamenti

Prima di procedere con la trattazione, vorrei dedicare qualche riga a tutti coloro che mi sono stati vicini in questo percorso di crescita personale e professionale.

Ringrazio il mio relatore, Leporati Alberto, che in questi mesi di lavoro, ha saputo guidarmi, con suggerimenti pratici, nelle ricerche e nella stesura dell'elaborato. Un ringraziamento speciale al mio correlatore Nicodemo Raffaele, per la sua immensa pazienza, per i suoi indispensabili consigli, per le conoscenze trasmesse durante tutto il periodo di stage. Un grazie di cuore a tutti i membri di Efebia, in cui ho svolto un tirocinio formativo, per l'ospitalità e per le skills acquisite sul campo.

Ringrazio infinitamente i miei genitori che mi hanno sempre sostenuto, appoggiando ogni mia decisione, fin dalla scelta del mio percorso di studi.

Ringrazio i miei compagni di corso con i quali ho condiviso momenti indimenticabili, augurandogli un futuro brillante e prospero, con la speranza che presto o tardi ci ritroveremo ancora. Non posso non ringraziare i miei amici che in modi diversi, attraverso parole, gesti, messaggi mi hanno supportato in questo periodo della mia vita.

Infine ringrazio tutte quelle persone che da sempre mi sostengono nella realizzazione dei miei progetti.

Sommario

In un mondo improntato sempre più alla globalizzazione e al digitale, la tecnologia ha pervaso ogni aspetto della nostra vita. Malgrado ciò, non è ancora stato creato un sistema di voto elettronico che soddisfi i requisiti richiesti per il corretto svolgimento delle elezioni politiche. In questo momento di crisi mondiale dovuta alla pandemia, è necessario sviluppare strumenti tecnologici che permettano il normale svolgimento delle attività quotidiane.

Negli ultimi anni sta emergendo una nuova tecnologia, la blockchain, che grazie alle sue proprietà di sicurezza, trasparenza e immutabilità, potrebbe essere l'innovazione più grande dopo Internet. In questo elaborato sarà mostrato come attraverso l'uso della blockchain e delle dimostrazioni zero knowledge sia possibile progettare un sistema di voto elettronico che assicuri trasparenza, sicurezza e affidabilità.

Inizialmente verranno spiegate la tecnologia blockchain e le dimostrazioni zero-knowledge. Dopo la descrizione di alcuni sistemi di voto elettronico basati su blockchain, si illustreranno le scelte implementative che hanno portato alla definizione di un protocollo di voto elettronico da remoto. In seguito si analizzeranno i risultati ottenuti dall'utilizzo del sistema di voto elettronico e si discuteranno i limiti di questa soluzione.

Indice

1	Introduzione	1
1.1	Obiettivo della tesi	1
1.2	Cos'è la blockchain	1
1.3	Voto elettronico	3
1.4	Struttura della tesi	4
2	Bitcoin	5
2.1	Crittografia basata su curve ellittiche	6
2.2	Blocchi	7
2.3	Transazioni	8
2.4	Script	9
2.5	Proof of Work	10
3	Dimostrazioni zero knowledge	13
3.1	Zero knowledge proof	13
3.1.1	Arguments of knowledge	14
3.1.2	Caverna di Ali Baba	16
3.2	ZK-SNARKs in Zcash	17
3.2.1	Quadratic span program	18
3.2.2	ZK-SNARKs nel dettaglio	19
3.2.3	ZK-SNARKs per il problema QSP	20
3.2.4	Zcash ceremony	22
3.3	Bulletproof in Monero	23
3.3.1	Prerequisiti	23
3.3.2	Bulletproof	26
3.4	Confronto tra ZK-SNARKs e Bulletproof	27
4	Esempi di voto elettronico	28
4.1	Votebook	29
4.2	Votewatcher	30
4.3	Polys	30
4.4	Crypto-Voting	31
4.5	Agora	32
4.5.1	Bulletin Board	33
4.5.2	Cotena	34
4.5.3	Valeda	34
4.5.4	Casting	34
4.6	I-voting	36
4.6.1	KSI blockchain	36
4.7	Voatz	38
4.8	VoteCoin	39

5	Sistema di voto basato su Monero	40
5.1	Monero	41
5.1.1	Crittografia	41
5.1.2	Transazione	44
5.1.3	Ring Signature	45
5.2	Sistema di voto	48
5.3	Implementazione	51
5.3.1	Avvio	52
5.3.2	Modifiche apportate	53
5.3.3	Ulteriori modifiche necessarie	55
5.3.4	Esperimento	57
6	Risultati ottenuti	70
6.1	Analisi di sicurezza	70
6.2	Problemi e criticità	77
6.3	Considerazioni finali	78
7	Conclusioni	80
7.1	Valutazioni finali e conclusioni	80
7.2	Possibili sviluppi futuri	81

Elenco delle figure

1	Dimostrazione geometrica dell'addizione su curva ellittica	6
2	Versione semplificata della blockchain di Bitcoin	8
3	Esempio di Merkle tree	8
4	Esempio di una transazione Bitcoin	10
5	Caverna di Ali Baba che rappresenta il protocollo zero-knowledge	16
6	Crypto-voting system	32
7	Interazione tra il layer Cotenà e Bitcoin	35
8	Esempio di KSI blockchain	37
9	Meccanismo della ring signature	45
10	Architettura del sistema di voto	58
11	Dashboard del servizio AWS	59
12	Istanze e AMI	59
13	Esempio di script per l'esecuzione di un nodo	61
14	Homepage di V-Monero	64
15	Errore in caso di codice fiscale errato	65
16	Errore in caso di codice fiscale già utilizzato	66
17	Pagina contenente il quesito sottoposto	66
18	Pagina dei ringraziamenti	67
19	Verifica del transaction Id	68
20	Errore nella verifica del transaction Id	68
21	Homepage al termine della votazione	69
22	Pagina dei risultati	69
23	Risultati della votazione effettuata con V-Monero	70
24	Errore generato da una transazione che utilizza un output non spendibile	72

Elenco delle tabelle

1	Confronto tra ZK-SNARKs e Bulletproof	28
2	Composizione di un indirizzo nella blockchain di Monero	43
3	Proprietà di un sistema di voto	49
4	Frequenze degli output appartenenti ai wallet analizzati	73
5	Riassunto delle modalità con cui V-Monero rispetta le proprietà di un sistema di voto	74

1 Introduzione

Questo lavoro nasce dall'opportunità di stage presso la startup Efebia, alla fine del 2019. L'idea ispiratrice con cui nasce Efebia è la creazione di un protocollo di voto elettronico basato su blockchain.

In occasione del referendum tenutosi nel 2017 per l'autonomia delle regioni, la regione Lombardia decise di acquistare migliaia di *voting-machine*, ovvero dispositivi elettronici per esprimere il proprio voto. Questa scelta generò molte discussioni a causa delle ingenti spese e della scarsa efficienza del sistema. Sebbene si utilizzarono dei dispositivi elettronici, il processo si svolse tutto offline. I voti vennero scaricati su chiavette che a loro volta furono raccolte per il conteggio finale dei voti [1]. Questo evento, unito alla problematica della falsificazione dei voti per corrispondenza spinsero i futuri fondatori di Efebia, a pensare di implementare un sistema di voto elettronico.

1.1 Obiettivo della tesi

L'obiettivo di questo elaborato è mostrare come grazie all'utilizzo di Monero, una blockchain attenta all'anonimato, sia possibile costruire un sistema di voto elettronico che garantisca tutte le proprietà necessarie per lo svolgimento di un corretto protocollo di voto.

In questo periodo, dove sono proibiti assembramenti e il distanziamento sociale è fondamentale, persone e aziende hanno la necessità di strumenti tecnologici per svolgere attività che normalmente venivano svolte in presenza. Per questo motivo, questa ricerca assume ancora più valore.

La blockchain viene spesso descritta come una tecnologia dirompente che può rivoluzionare diversi settori, grazie alle sue proprietà di sicurezza, trasparenza e immutabilità. Nel 2017 l'Unione europea ha pubblicato un documento [2] nel quale vengono spiegati i diversi ambiti in cui questa tecnologia potrebbe essere impiegata. Proprio in questo documento, un capitolo è dedicato alla creazione di sistemi di voto elettronico.

Inoltre negli ultimi anni il numero di progetti basati su blockchain ha avuto una forte crescita in tutto il mondo [3], a testimonianza del crescente interesse riguardo questa tecnologia.

1.2 Cos'è la blockchain

La parola blockchain viene spesso utilizzata in maniera inappropriata quando si parla di innovazione e nuove tecnologie. Il termine è stato coniato successivamente all'implementazione e alla diffusione di Bitcoin. Infatti nel whitepaper scritto dal creatore, Satoshi Nakamoto, per spiegare la struttura dati che viene memorizzata, si fa riferimento ad una catena a blocchi, letteralmente "*block of chain*" [4].

Dopo Bitcoin sono stati implementati molti altri protocolli che in maniera più o meno simile mirano a costruire un network decentralizzato che garantisca le proprietà di sicurezza, affidabilità ed immutabilità. In maniera formale, si definisce blockchain un insieme di tecnologie e protocolli che richiede quattro elementi:

1. rete peer to peer;
2. registro distribuito;
3. utilizzo della crittografia;
4. algoritmo di consenso distribuito.

Una rete peer to peer (spesso abbreviata P2P), è una rete caratterizzata dall'assenza di una struttura gerarchica tipica dei sistemi client/server e nella quale ciascun nodo può operare come client o come server a seconda delle circostanze. Storicamente le reti peer to peer sono legate al fenomeno del *file sharing*, ovvero la condivisione di file all'interno di una rete.

Il termine registro distribuito indica che ciascun nodo mantiene una copia locale del database. Questa è la prima proprietà che rende la blockchain una rete decentralizzata, quindi senza *single point of failure*. Questa caratteristica rende la blockchain un sottoinsieme della famiglia delle *distributed ledger* (DLT). Solitamente i dati vengono memorizzati in transazioni raggruppate in blocchi. Ciascun blocco contiene il riferimento al blocco precedente in modo da costruire una catena. Quando viene confermato un nuovo blocco, ciascun nodo aggiorna il proprio database in locale.

Nella tecnologia blockchain si fa grande utilizzo della crittografia, in particolare di crittosistemi asimmetrici e *funzioni one-way*. Una funzione one-way è una funzione per cui dato l'output è impossibile risalire all'input. Tipicamente le funzioni one-way utilizzate nella blockchain sono funzioni di hash.

In assenza di un'autorità centrale o di una terza parte, l'algoritmo di consenso definisce chi certifica la correttezza di un blocco e di conseguenza anche delle transazioni al suo interno. Per stabilire se un blocco è valido, in una rete P2P non si può utilizzare l'identità dei nodi poiché è possibile imitare qualcun'altro oppure gestire molteplici nodi. Per questo motivo non è applicabile una qualsiasi forma di democrazia diretta. Il consenso in un sistema distribuito è difficile poiché:

- i nodi possono essere difettosi per malfunzionamenti tecnici o attività maliziose arbitrarie;
- i nodi non sono tutti connessi a coppie;
- esiste una latenza nella comunicazione tra i nodi;

- non esiste una nozione di tempo globale.

Fisher, Lynch e Paterson hanno dimostrato [5] che in una rete asincrona non è possibile ottenere il consenso distribuito neanche sul valore di una semplice variabile booleana. Quindi il consenso distribuito è teoricamente impossibile da raggiungere. Questo perché siamo in un contesto cosiddetto bizantino, ovvero con nodi malevoli.

Il problema dei generali bizantini è la metafora con cui si descrive il problema del consenso nei sistemi distribuiti. Si immagini la situazione in cui, durante l'assedio di una città, diversi generali devono decidere se attaccare o ritirarsi. Essi possono comunicare solo attraverso messaggeri. Ci sono dei traditori tra di loro che vanno contro la strategia dell'esercito ed i generali devono accordarsi all'unanimità se attaccare oppure no. Il successo si ottiene se i generali "leali" si accordano sulla strategia, qualunque essa sia. Grazie all'utilizzo della *proof of work*, Bitcoin riesce ad ottenere un consenso dal punto di vista pratico (PBFT), anche nel caso di contesto bizantino. Il consenso distribuito è sempre implicitamente probabilistico.

Come espresso da Buterin [6], è possibile suddividere le applicazioni blockchain in tre categorie:

- public blockchains: ovvero blockchain in cui chiunque può leggere e inviare transazioni, oltre a partecipare al protocollo di consenso;
- consortium blockchains: cioè blockchain dove il processo di consenso viene eseguito da un sottoinsieme preselezionato di nodi;
- private blockchains: cioè blockchain dove i permessi di scrittura vengono mantenuti da un'autorità centrale. I permessi di lettura possono essere pubblici o limitati in maniera arbitraria.

Sempre secondo Buterin, una blockchain privata può essere descritta più accuratamente come un sistema centralizzato tradizionale con l'aggiunta di un livello di verificabilità crittografica [6].

1.3 Voto elettronico

Il voto basato su carta è il sistema di voto più utilizzato al mondo poiché permette facilmente di verificare che le proprietà di trasparenza e segretezza siano rispettate. Infatti è abbastanza semplice controllare che il voto venga effettuato segretamente rispettando il protocollo in uso. D'altra parte il costo per effettuare un'elezione è abbastanza alto. Bisogna considerare il costo delle schede, del necessario per organizzare i seggi, oltre al costo del trasporto. Per un'elezione basata su carta è necessario coinvolgere degli amministratori delle elezioni e dei dipendenti che li affianchino. Essi sono sensibili a frodi e corruzioni, poiché hanno il compito di conteggiare i voti e stabilire se un voto è valido. Inoltre il processo è vulnerabile all'errore umano. Infine l'accessibilità al voto

è limitata. Le persone spesso sono costrette ad effettuare diverse ore di coda per esprimere il proprio voto. Le persone con disabilità hanno grossi problemi ad effettuare un voto su carta e il voto per email o corrispondenza non è sicuro.

Alcuni paesi hanno sperimentato l'utilizzo di electronic voting machine (EVM) non in remoto. Questi sistemi prevengono alcuni problemi come il costo del materiale, la correttezza del conteggio del voto ma ne introducono altri, come il costo dell'acquisto e dell'aggiornamento dei dispositivi, l'assistenza tecnica e la formazione del personale. Questi sistemi sono intrinsecamente opachi e non c'è modo di verificare che il voto sia stato correttamente registrato. Inoltre la maggior parte delle EVM sono soluzioni proprietarie e quindi il codice non è open source e il fornitore del servizio è responsabile dell'integrità del sistema. Le EVM sono dispositivi hardware che possono essere alterati tramite l'immissione di codice malevolo. Se sono collegati a Internet sono più vulnerabili a problemi di sicurezza. Infine le EVM hanno un ciclo di vita da 10 a 20 anni. Ad esempio durante le elezioni in Virginia, i dispositivi di WinVote sono risultati vulnerabili poiché utilizzavano una scheda wireless che impiegava uno standard crittografico obsoleto [7].

Le prime dissertazioni scientifiche riguardo la creazione di un protocollo di voto elettronico risalgono ai primi anni 2000 e sebbene in alcune parti del mondo siano state fatte alcune sperimentazioni, non esiste ancora un sistema considerato univocamente sicuro ed affidabile.

1.4 Struttura della tesi

Nel prossimo capitolo verrà brevemente presentato Bitcoin, con l'obiettivo di introdurre alcune nozioni fondamentali per comprendere al meglio il funzionamento della blockchain. Nel capitolo 3 verranno esposte le dimostrazioni zero knowledge, in particolare quelle implementate in Zcash e Monero. Nel capitolo 4 verranno illustrati alcuni esempi di sistemi di voto che utilizzano la blockchain. Nel capitolo 5 si analizzerà il funzionamento di Monero, quali modifiche sono state necessarie per utilizzarlo in un protocollo di voto e come si è svolto l'esperimento. Nel capitolo 6 si commenteranno i risultati ottenuti e le problematiche riscontrate.

2 Bitcoin

Bitcoin nasce dall'esigenza di creare una valuta digitale decentralizzata senza alcun banchiere centrale che convalidi le transazioni. Il motto è “be your own bank”. Infatti, ciascun partecipante mantiene una copia locale della blockchain, contenente tutte le transazioni.

Immaginando l'esistenza di una moneta digitale, essa viene creata da un'autorità centrale, firmata con la chiave privata o codice seriale dalla banca centrale e assegnata ad una persona che può spenderla. Dato che in Bitcoin non esiste un'autorità centrale, è il mittente a firmare le transazioni. Ogni volta che la banconota cambia proprietario, essa viene firmata dal mittente. In questo modo è possibile ricostruire la storia della banconota a partire dalla sua creazione. Questa caratteristica è fondamentale per evitare il problema della doppia spesa. Si definisce problema della doppia spesa la situazione in cui un utente, ad esempio Alice, tenta di trasferire la stessa somma di denaro sia a Bob che a se stessa, o ad un terzo utente. Quando esiste un'autorità centrale, il codice seriale o la firma con la chiave privata risolve questo problema. Invece per la rete Bitcoin entrambe le transazioni sarebbero valide. Perciò è necessario che qualcuno verifichi e validi le transazioni. I *miner* sono quei nodi che si occupano di validare le transazioni.

Le transazioni sono organizzate in blocchi, singole unità per il lavoro di validazione. La suddivisione in blocchi favorisce una validazione più veloce. Per validare un blocco, un miner deve generare l'hash del blocco. Come è stato spiegato nell'introduzione, in una rete distribuita asincrona è possibile ottenere il consenso distribuito pratico tramite la *proof of something*. Bitcoin utilizza la proof of Work, abbreviata in PoW. I partecipanti devono effettuare del “lavoro” computazionale per validare un blocco. Il lavoro consiste in un puzzle crittografico, che artificialmente accresce il costo della transazione. In questo modo, l'abilità di verifica dipende dalla potenza computazionale e non dal numero di identità potenzialmente false. Questa scelta è dovuta al fatto che in una rete P2P è più facile impersonificare un gran numero di nodi piuttosto che ottenere la maggior parte della potenza computazionale. Il puzzle utilizzato nella PoW consiste nel trovare un *nonce*, tale per cui l'hash $h(\text{prev_hash} || \{Tx, \dots, Tx\} || \text{nonce})$ sia uguale o minore di un certo valore target. Quando un partecipante trova il nonce, il blocco con il rispettivo nonce viene distribuito nella rete e i partecipanti aggiornano la loro blockchain locale. Risolvere il puzzle è computazionalmente difficile.

Bitcoin utilizza la funzione di hash SHA256. L'unico modo per trovare la soluzione corretta è calcolare la funzione provando tutti i possibili valori. Più il valore target è piccolo, ovvero richiede più zero iniziali, più è difficile risolvere il puzzle. Si utilizza una funzione di hash in modo che sia difficile risolvere il puzzle crittografico mentre è estremamente facile verificarne la correttezza.

Il protocollo Bitcoin non è facilmente modificabile. Per essere modificato richiede che l'intero network si accordi sulle modifiche da apportare. Alcune volte una parte degli

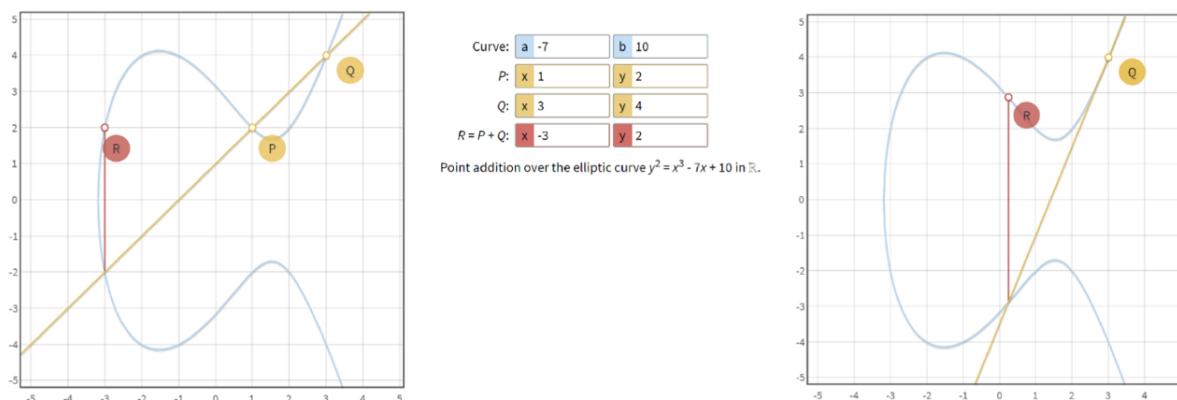


Figura 1: Dimostrazione geometrica dell'addizione su curva ellittica

utenti decide di effettuare cambiamenti al protocollo generando un *hard fork*, come ad esempio nel caso di Bitcoin Cash. In questi casi si sviluppano due catene che continuano a crescere in maniera parallela ed indipendente.

2.1 Crittografia basata su curve ellittiche

Una curva ellittica è una curva algebrica definita su un campo K . Ogni curva ellittica può essere scritta come una curva algebrica piana, definita dall'equazione di Weierstrass:

$$y^2 = x^3 + ax + b, \quad \text{con } a, b \in K \wedge \Delta \neq 0$$

Le curve ellittiche sono spesso utilizzate nell'ambito della crittografia.

La somma su una curva ellittica ha un significato geometrico. Presi due punti P e Q sulla curva, la somma R si calcola tracciando la retta passante per i punti P e Q . Questa retta intercetta un altro punto sulla curva. Il simmetrico a questo punto è R , il risultato dell'addizione. Questa osservazione geometrica vale anche se P e Q sono coincidenti e quindi la retta è tangente (figura 1).

Dato che le curve ellittiche sono simmetriche rispetto all'asse delle ascisse, per ogni elemento appartenente al gruppo esiste un punto con ascissa uguale e ordinata opposta. Per questo motivo è possibile comprimere la rappresentazione dei punti alla rappresentazione della x più 1 bit che determina se y è positivo o negativo.

Come nel whitepaper di Bitcoin, la legge di gruppo viene definita addizione. Con

l'operazione di addizione, la curva ellittica è un gruppo commutativo, dove il punto all'infinito (una nozione che deriva dalla Geometria proiettiva) è l'elemento neutro. Calcolare la moltiplicazione di un punto, $R = mQ$, è una notazione comoda per indicare una ripetizione dell'applicazione dell'addizione ($m \in \mathbb{N}$). La moltiplicazione su curva ellittica non è un'operazione binaria sull'insieme dei punti della curva; l'unica operazione binaria del gruppo sulla curva ellittica è l'addizione. Inferire m da R e Q è computazionalmente difficile. Inoltre, è semplice calcolare la moltiplicazione, mentre è estremamente difficile calcolare la divisione. Per questo motivo la moltiplicazione su curva ellittica è una funzione one-way. Partendo da R e Q , l'unico modo per conoscere m è tentare iterativamente. Nella sua notazione moltiplicativa, questo problema viene definito come problema del logaritmo discreto.

Bitcoin utilizza la curva ellittica secp256k1 (curva di Koblitz). Secp256k1 è la curva ellittica $y^2 = x^3 + 7$, utilizzata anche nell'algoritmo ECDSA e implementata secondo gli standard per l'efficienza crittografica [8]. In Bitcoin questa curva viene utilizzata per generare la chiave pubblica partendo da quella privata. La chiave pubblica è un punto R sulla curva ellittica mentre la chiave privata è il valore intero m che è stato moltiplicato per il punto G (corrispondente a Q nell'esempio precedente).

In Bitcoin, le chiavi sono codificate in base 58. A seconda del tipo di rappresentazione utilizzata per la chiave pubblica (compressa o non compressa), si ottengono indirizzi e chiavi differenti. Queste differenti versioni si riconoscono dal prefisso degli indirizzi.

Memorizzare le chiavi su file e quindi su un device non è sicuro. Per questo motivo è stato inventato il concetto di *wallet*, ovvero un software, hardware o un pezzo di carta che permetta di gestire le chiavi private, pubbliche e gli indirizzi. Per spendere bitcoin (BTC) in maniera sicura, un wallet ha bisogno della chiave privata a cui sono associati i bitcoin, e le informazioni sulla public blockchain. È possibile collegarsi alla rete Bitcoin, direttamente, eseguendo un *full node*, oppure nel caso di un *light client* ci sia affida ad un trusted server. Un node client è connesso continuamente alla rete Bitcoin, e memorizza l'intera blockchain (circa 300 GB nel 2020). Un nodo riceve e trasmette ogni transazione e ogni blocco, effettuandone la validazione. Affinchè sia veramente affidabile, il nodo deve rimanere connesso alla rete senza interruzioni. Questa è la soluzione più sicura. Al contrario un light client memorizza solo i *block header* (circa 52MB nel 2018). Esso si connette al trusted node e invia transazioni solo quando ne ha necessità. Per verificare la validità e la conferma delle transazioni, si affida completamente ai full node.

2.2 Blocchi

La struttura dati impiegata in Bitcoin è una lista concatenata di blocchi, conosciuta meglio con il nome di catena di blocchi, o blockchain. Si definisce altezza della blockchain

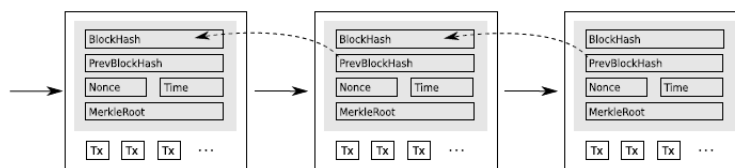


Figura 2: Versione semplificata della blockchain di Bitcoin

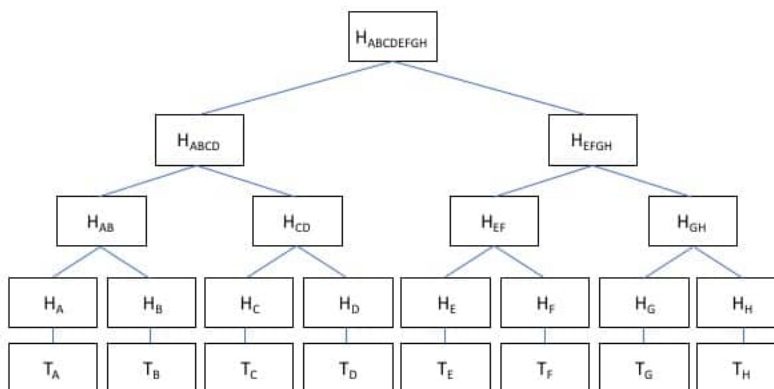


Figura 3: Esempio di Merkle tree

il numero di blocchi nella catena. Un blocco è formato da un block header, contenente i metadati del blocco, e da una lista di transazioni (figura 2). Con l'aumento del numero di transazioni, sono necessari 300 gigabyte per scaricare l'intera blockchain. Per contenere la grandezza delle transazioni e ridurre lo sforzo computazionale per verificare la convalida di una transazione, Bitcoin offre un meccanismo semplificato per la verifica dei pagamenti, SPV, basato su Merkle Tree.

Il Merkle Tree è una struttura dati in cui le transazioni di un blocco sono foglie di un albero binario e ad ogni livello superiore viene eseguito l'hash dei nodi figli, come mostrato nell'esempio della figura 3. La radice viene inclusa nel block header. In questo modo l'hash tree permette la verifica delle transazioni senza la necessità di ottenere la copia locale di tutte le transazioni.

2.3 Transazioni

Bitcoin utilizza la crittografia asimmetrica basata su curve ellittiche. Per inviare e ricevere bitcoin, un utente deve creare un wallet, formato da una coppia di chiavi, una pubblica ed una privata. Esistono diversi tipi di wallet. Per quello più semplice, l'indirizzo del wallet si ottiene tramite la funzione SHA256 della chiave pubblica.

Una transazione è composta da un campo TxId, ovvero l'hash che identifica univocamente la stessa, una lista di input e una lista di output. Per ogni input viene specificato l'hash della transazione che lo ha generato, il campo index, cioè la sua posizione nella lista degli output al momento della creazione e il campo scriptSig, contenente la firma del mittente. Per ogni output viene specificato nel campo value l'ammontare di coin, e uno script che definisce come spendere questi coin. Nella figura 4 viene mostrato un esempio di una transazione Bitcoin.

Durante una transazione, gli input vengono consumati e vengono generati nuovi output. Gli output di una transazione possono essere categorizzati come *unspent transaction output*, UTxO, se non sono riferiti al momento a nessuna transazione seguente, oppure *spent transaction output*, STxO. Un input non può essere speso parzialmente. Per questo motivo per ricevere il resto, uno degli UTxO specificati appartiene al mittente della transazione. La somma degli input deve essere maggiore o uguale alla somma degli output. La differenza viene trasferita al miner che validerà il blocco contenente la transazione e prende il nome di *transaction fee*. Logicamente maggiore è la transaction fee e più velocemente la transazione sarà validata.

Esistono speciali transazioni che sono chiamate *coinbase transaction*. Queste transazioni sono incluse dal miner durante la validazione di un blocco. In questo modo vengono generati nuovi bitcoin e il miner viene ricompensato per aver validato il blocco. Per spendere il contenuto di una coinbase transaction bisogna aspettare che siano stati validati almeno 100 blocchi.

2.4 Script

Una transazione Bitcoin non è semplicemente un trasferimento di valore. Infatti come è stato detto nel paragrafo precedente, ad ogni output viene associato uno script contenente un puzzle matematico. Per spendere il transaction output, il corrispondente input deve risolvere il puzzle codificato con il Bitcoin Script language. Il Bitcoin Script language è un semplice linguaggio stateless, non Turing-completo. Questa limitata flessibilità è stata adottata per evitare vulnerabilità. In questo linguaggio:

- non esistono cicli né flussi di controllo complessi;
- è assicurata la terminazione;
- non esistono variabili e i calcoli vengono eseguiti sullo stack.

Affinchè un UTxO sia spendibile, il risultato dello script deve essere TRUE. Lo script più comune è il P2PKH (*Pay to public key hash*). Esso verifica che il mittente che firma l'input sia il destinatario del transaction output utilizzato.



Figura 4: Esempio di una transazione Bitcoin

2.5 Proof of Work

In una rete P2P non si può utilizzare l'identità dei nodi poiché è possibile imitare qualcun altro o gestire molteplici nodi. Per questo motivo non si può sostituire il banchiere centrale con una qualsiasi forma di democrazia diretta.

In un sistema distribuito e decentralizzato, l'algoritmo di consenso deve considerare che:

- ogni nodo può distribuire nuove transazioni;
- le transazioni vengono organizzate in blocchi;
- a turno si sceglie in maniera casuale un nodo che propone un blocco;
- i nodi accettano un blocco implicitamente se costruiscono un nuovo blocco a partire da questo.

L'algoritmo di consenso principalmente determina come viene scelto il nodo che propone un blocco. Esistono diversi tipi di algoritmi di consenso e possono essere suddivisi in due categorie, proof of work o proof of stake. La proof of stake sceglie il nodo in base al numero di coin posseduti. Invece la proof of work si basa sul dispendio di risorse computazionali che hanno un corrispettivo valore reale. Infatti è costoso comprare e mantenere hardware predisposto alle computazioni. Per questo motivo la PoW viene incentivata dalle transaction fee e dalle coinbase transaction, altrimenti non sarebbe conveniente per i miner.

In generale si considera la PoW più sicura poiché è costoso mantenere più ramificazioni della stessa blockchain o qualsiasi altro tipo di disaccordo nel consenso. Inoltre, qualora un attaccante controllasse la potenza computazionale necessaria, l'attacco avrebbe comunque un costo in termini energetici. Bitcoin utilizza la proof of work.

Esistono due tipi di proof of work:

- protocollo challenge and response;
- protocollo solution-verification, come in Bitcoin.

In Bitcoin l'obiettivo della proof of work è quella di trovare un nonce tale per cui l'hash $h(\text{prev_hash} || \{Tx, \dots, Tx\} || \text{nonce})$ appartenga ad un intervallo piccolo rispetto allo spazio degli output. Una maggiore potenza computazionale permette di aumentare la probabilità di essere il primo ad ottenere la soluzione.

Per ragioni di stabilità e tempo di attesa per la validazione delle transazioni, si è deciso di aggiustare l'intervallo obiettivo ogni 2016 blocchi (2 settimane), per permettere la verifica di un blocco in un tempo costante, approssimativamente ogni 10 minuti.

In qualsiasi momento la catena di riferimento è quella con la difficoltà cumulativa più alta e quindi quella più lunga. Durante la validazione di un blocco, possono generarsi storie alternative, ovvero più blocchi che sono collegati ad uno stesso blocco. I blocchi minati che non appartengono alla catena principale sono detti blocchi orfani. I blocchi orfani sono abbastanza comuni. Un'euristica spesso utilizzata suggerisce di aspettare sei conferme prima di considerare la transazione registrata. Il successo di un attacco di doppia spesa decresce esponenzialmente con l'aumentare delle conferme di un blocco.

Per incentivare i partecipanti ad essere onesti e minare i blocchi, Bitcoin prevede due tipi di ricompense: transactions fees e mining. L'operazione di mining consiste, come si è detto sopra, nell'aggiungere un nuovo blocco alla catena. L'operazione di mining permette di inserire tra le transazioni del blocco una transazione di *block reward* a vantaggio del minatore. In questo modo vengono generati nuovi bitcoin. Le transaction fee sono state spiegate nel paragrafo 2.3.

I bitcoin sono in numero finito, circa $21 \cdot 10^6$. La ricompensa iniziale era di 50 BTC. Ogni 210 000 blocchi, circa 4 anni, la ricompensa si dimezza. L'unità minima di bitcoin, conosciuta come satoshi, corrisponde a 10^{-8} BTC ed è previsto che verrà minata attorno al 2140.

Sebbene la ricompensa continui a diminuire e la difficoltà ad aumentare, rimane comunque conveniente fare mining poiché il valore di bitcoin, considerato su scala logaritmica, continua a crescere.

Per alterare la blockchain di Bitcoin è necessario un attacco con potenza computazionale pari o superiore al 51% della potenza utilizzata per incrementare la catena principale. Attualmente il network Bitcoin necessita di una potenza computazionale tale che nemmeno l'unione dei maggiori supercomputer del mondo o l'attacco di superpotenze mondiali può intaccare la sicurezza della rete. Questo rende Bitcoin il network transazionale più sicuro al mondo. Inoltre, un qualsiasi agente economico razionale è incentivato a minare piuttosto che attaccare la rete.

Anche in un ipotetico scenario d'attacco, non è possibile modificare il protocollo, rubare coin da un indirizzo esistente o rimuovere le transazioni dalla rete. Un attacco potrebbe:

- prevenire che le transazioni vengano confermate;
- sferrare un attacco DoS nei confronti di uno specifico indirizzo;
- riscrivere transazioni recenti che sono state confermate, effettuando attacchi di doppia spesa;
- ridurre la fiducia nei bitcoin, innescando un crollo dei prezzi.

Per un singolo minatore è estremamente remunerativo risolvere la PoW, ma capita estremamente raramente. Quindi sempre più miner si sono raggruppati in *mining pool* dove si genera il blocco congiuntamente. Ciascun partecipante ricerca il nonce valido su uno spazio di nonce ristretto. In caso di successo, la ricompensa viene condivisa. In questo modo le ricompense sono minori ma più frequenti. Esistono molteplici funzioni di pagamento per condividere i profitti in una mining pool.

È capitato che più blocchi consecutivi siano stati validati dallo stesso mining pool. In alcuni casi un mining pool ha superato il 51% di potenza computazionale, o comunque l'unione di più pool potrebbe superarlo. È nell'interesse di tutti i miner mantenere l'ecosistema distribuito. La soluzione più semplice è quella che i miner in prima persona cambino mining pool per mantenere la potenza computazionale distribuita. Un'altra possibilità sono i mining pool decentralizzati, dove i diversi miner si sfidano per comporre una catena di blocchi detta *share chain*. I blocchi di questa catena sono gli stessi che verrebbero proposti alla blockchain di Bitcoin ad eccezione di una difficoltà obiettivo minore. Per di più ogni blocco aggiunto nella share chain, per essere valido, deve comprendere un pagamento ai diversi miner che hanno contribuito alla creazione della catena. La composizione di questa catena continua fino a quando non viene trovato un blocco che soddisfi la difficoltà di Bitcoin. Quindi se un mining pool decentralizzato ottenesse il 51%, questa potenza non potrebbe essere sfruttata da un coordinatore centrale.

3 Dimostrazioni zero knowledge

In matematica le prove sono viste tradizionalmente come oggetti statici. Una “prova” viene spesso rappresentata come una sequenza fissa di affermazioni assiomatiche o derivate da precedenti affermazioni tramite regole di derivazione. Al contrario, le *zero knowledge proof* sono stabilite attraverso un processo interattivo.

Introduciamo alcuni termini che solitamente vengono utilizzati nei *proof system*. Il *prover* è l'entità che fornisce la prova, mentre il *verifier* è colui che attivamente verifica la veridicità di una particolare affermazione [9]. In un sistema interattivo, la prova viene generata dall'interazione tra il prover e il verifier.

3.1 Zero knowledge proof

Le zero-knowledge proof sono state ideate per la prima volta nel 1985, in un articolo scritto da Goldwasser, Micali e Rackoff [10]. Nell'articolo si parla di una gerarchia di interactive proof system e viene definita la nozione di “*knowledge complexity*”, una misura della conoscenza trasferita dal prover al verifier. Da qui nasce il termine zero knowledge.

In una successiva pubblicazione [11] è stato dimostrato che partendo da un qualsiasi problema nella classe di complessità NP è possibile generare una zero knowledge proof.

In crittografia, una dimostrazione zero knowledge o un protocollo zero-knowledge è un protocollo attraverso il quale un prover può convincere un verifier di conoscere un valore segreto x , senza rivelare alcuna informazione riguardo al segreto, ad eccezione dell'affermazione di conoscere il valore x . L'essenza delle zero-knowledge proof è il fatto che per dimostrare di conoscere una certa informazione non è necessario rivelarla, la sfida è quella di dimostrare di conoscere qualcosa senza rivelare alcuna informazione su di essa.

Una zero-knowledge proof deve soddisfare queste tre proprietà:

- **completeness:** se l'affermazione è vera, allora un verifier onesto, cioè uno che segue il protocollo correttamente, sarà convinto dell'affermazione data da un prover onesto;
- **soundness:** se l'affermazione è falsa, nessun prover malevolo può convincere un verifier onesto che essa sia vera, o meglio la probabilità di convincerlo può essere resa bassa a piacere;
- **zero-knowledge:** se un'affermazione è vera, nessun verifier può apprendere alcuna informazione oltre al fatto che l'affermazione è vera. In altre parole, conoscere semplicemente l'affermazione (non il segreto) è sufficiente per immaginare uno scenario che mostri che il prover conosce il segreto.

Le prime due proprietà appartengono ai più generali interactive proof systems.

Definizione 1 (Interactive Proof System)

Siano P e V una coppia di algoritmi interattivi. $\langle P, V \rangle$ è un interactive proof system per il linguaggio \mathbb{L} se V può solo fare computazioni in tempo polinomiale e

- **completeness:** per ogni $x \in \mathbb{L}$

$$P[\langle P, V \rangle(x) = 1] \geq 1 - \delta$$

- **soundness:** per ogni $x \notin \mathbb{L}$ e qualsiasi algoritmo B

$$P[\langle B, V \rangle(x) = 1] \leq \delta$$

dove $\langle P, V \rangle(x) = 1$ indica che la prova viene accettata. Invece δ è una quantità piccola a piacere. Nel caso in cui $\delta = 0$, si parla di perfect completeness e perfect soundness.

Esistono due famiglie di zero-knowledge proof. Si definisce **interactive zero knowledge proof** un protocollo che necessita dell'interazione tra gli attori del sistema, ovvero il prover interagisce con il verifier per dimostrare di conoscere il segreto riguardo all'affermazione. Se un osservatore terzo dovesse vedere lo scambio di messaggi, non sarebbe in grado di verificare la validità dell'affermazione.

Al contrario le **non interactive zero knowledge proof** vengono utilizzate per dimostrare un'affermazione ad un gruppo più ampio di attori. In questo caso non ci sono interazioni tra le parti, oppure sono minime.

3.1.1 Arguments of knowledge

Come viene descritto nel capitolo 2 dell'articolo [12], si definisce "argument of knowledge" una prova valida se il prover è computazionalmente limitato e sono valide le assunzioni di difficoltà computazionale.

Siano gli argomenti composti da tre algoritmi interattivi $(Setup, P, V)$, appartenenti alla classe PPT, ovvero algoritmi eseguiti in tempo polinomiale su una macchina di Turing probabilistica. In particolare il primo è l'algoritmo di *Setup* per la generazione della CRS (*common reference string*). Il secondo viene eseguito dal prover P mentre il terzo dal verifier V . Partendo da un input di lunghezza 1^λ (dove λ è un parametro di sicurezza), *Setup* produce una stringa di riferimento comune σ . La trascrizione prodotta da P e V quando interagiscono con gli input s e t , si definisce come $tr \leftarrow \langle P(s), V(t) \rangle$. Quindi $\langle P(s), V(t) \rangle = b$ dove b è un valore booleano che indica se il verifier accetta o rigetta la prova.

Sia $R \subset \{0, 1\}^* \times \{0, 1\}^* \times \{0, 1\}^*$ una relazione ternaria decidibile in tempo polinomiale.

Data σ , chiamiamo w un testimone per un'affermazione u se $(\sigma, u, w) \in R$; si definisce un linguaggio CRS-dipendente:

$$\mathbb{L}_\sigma = \{u \mid \exists w : (\sigma, u, w) \in R\}$$

come l'insieme delle affermazioni u che hanno un testimone w nella relazione R .

Definizione 2 (Argument of knowledge)

La tripla $(Setup, P, V)$ viene chiamata *argument of knowledge* per la relazione R se essa soddisfa le seguenti due definizioni.

Definizione 3 (Perfect completeness)

$(Setup, P, V)$ gode della proprietà di *perfect completeness* se per ogni algoritmo A eseguito in tempo polinomiale non uniforme:

$$P[(\sigma, u, w) \notin R \vee \langle P(\sigma, u, w), V(\sigma, u) \rangle = 1 \mid \sigma \leftarrow Setup(1^\lambda), (u, w) \leftarrow A(\sigma)] = 1$$

In altre parole il verifier accetta sempre la prova fornita dal prover oppure la tripla non appartiene alla relazione ternaria.

Definizione 4 (Computational witness-extended emulation)

La tripla $(Setup, P, V)$ ha un “emulazione witness-extended” se tra tutti gli algoritmi deterministici $P^*(\text{prover})$ eseguiti in tempo polinomiale esiste un emulatore ε eseguibile in tempo polinomiale tale che per tutte le coppe di “avversari” interattivi A_1, A_2 esiste una funzione trascurabile $\mu(\lambda)$ tale che:

$$\left| \frac{P[A_1(tr) = 1 \mid \sigma \leftarrow Setup(1^\lambda), (u, s) \leftarrow A_2(\sigma), tr \leftarrow \langle P^*(\sigma, u, w), V(\sigma, u) \rangle] - P[A_1(tr) = 1 \wedge (tr \text{ è accettato} \implies (\sigma, u, w) \in R) \mid \sigma \leftarrow Setup(1^\lambda), (u, s) \leftarrow A_2(\sigma), (tr, w) \leftarrow \epsilon^O(\sigma, u)]}{1} \right| \leq \mu(\lambda)$$

dove l'oracolo $O = \langle P^*(\sigma, u, s), V(\sigma, u) \rangle$ permette al verifier di ritornare ad uno specifico punto e da lì in avanti riprendere la verifica con un nuovo valore casuale. La variabile s può essere considerata come lo stato di P^* , includendo anche la componente casuale.

In sostanza, quando un prover produce un argomento σ che soddisfa il verifier con una certa probabilità, allora esiste anche un emulatore che con la stessa probabilità genera lo stesso argomento, con l'aggiunta di un testimone w . L'emulatore è in grado di ripercorrere tutte le interazioni tra prover e verifier e di riprendere la verifica con lo stesso stato interno del prover utilizzando un nuovo valore casuale deciso dal verifier. Ogni qualvolta P^* genera un argomento convincente nello stato s , può estrarre il testimone e così si ottiene l'argomento di knowledge w , tale che $(\sigma, u, w) \in R$. Un argomento di knowledge è zero-knowledge se non lascia trasparire alcuna informazione riguardo w ad eccezione di ciò che può essere dedotto dal fatto che $(\sigma, u, w) \in R$.

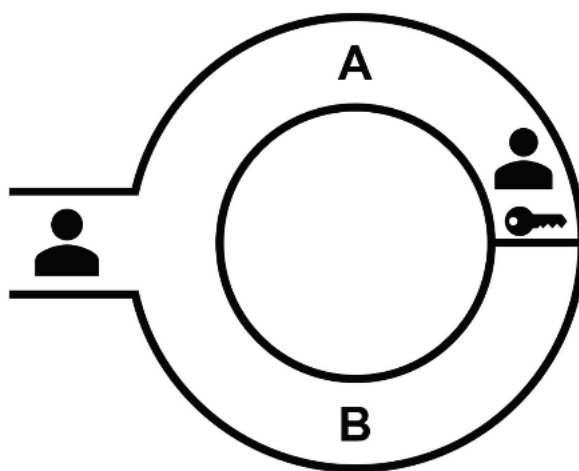


Figura 5: Caverna di Ali Baba che rappresenta il protocollo zero-knowledge

3.1.2 Caverna di Ali Baba

La storia della caverna di Ali Baba chiarisce l'idea di cosa sia una zero-knowledge proof. Essa è stata pubblicata da Jean-Jacques Quisquater e altri nel paper “How to Explain Zero-Knowledge Protocols to Your Children” [13].

Sia Peggy la persona che svolge il ruolo di prover e Victor quella che desidera verificare un'affermazione. Nella storia Peggy ha scoperto la parola segreta per aprire una porta magica nella caverna. La caverna è a forma di anello, con un'entrata da un lato; nella parte opposta è presente una porta magica che blocca la caverna. Victor vuole sapere se Peggy conosce la parola segreta. Peggy, essendo una persona riservata, non vuole rivelare la parola segreta a Victor e non vuole far sapere al mondo di possedere questa conoscenza (figure 5).

Le due strade all'ingresso della caverna vengono chiamate A e B. Victor attende all'ingresso della caverna mentre Peggy s'incammina lungo una delle due strade. A Victor non è permesso sapere quale strada ha intrapreso Peggy. Allora Victor, sceglie in maniera casuale tra A e B, entra nella caverna e urla il nome della strada dalla quale Peggy deve fare ritorno. Se Peggy conosce la parola segreta, ritornerà sempre dalla strada richiesta e utilizzerà la parola magica per aprire la porta segreta solo quando sarà necessario. Supponendo che essa non conosca la parola segreta, potrà ritornare dalla strada richiesta solo se aveva imboccato quella medesima strada. Ripetendo questa prova per venti volte, la probabilità che Peggy indovini anticipatamente la strada da cui ritornare senza dover conoscere la parola segreta è molto bassa: $\left(\frac{1}{2}\right)^{20} \simeq 0,000095\%$

Supponiamo ora che Victor indossi una telecamera. Un osservatore esterno che vede la registrazione potrebbe comunque pensare che Peggy e Victor abbiamo orchestrato tutto precedentemente. Immaginiamo allora che Victor lanci una moneta per decidere quale strada scegliere. Il lancio della moneta registrato dalla telecamera sarebbe probabilmente convincente per chiunque guardasse la registrazione in seguito. In questo modo Victor potrebbe convincere il mondo in generale che Peggy ha quella conoscenza, contrariamente ai desideri dichiarati da essa. Inoltre il lancio di una moneta generalmente viene simulato tramite un generatore di numeri pseudo-casuali che segue un pattern noto solo al proprietario. Quindi Victor e Peggy potrebbero comunque colludere per dimostrare l'esperimento.

Peggy può comunque provare di conoscere la parola segreta in un singolo tentativo. Se essi entrano insieme nella caverna e Victor vede Peggy partire dalla strada A e tornare da B non ci sono dubbi sul fatto che Peggy conosca la parola segreta. Allo stesso tempo registrando l'esperimento, Victor può convincere chiunque che Peggy conosce la parola magica, ed essa non avrebbe più il controllo su chi conosce il suo segreto.

3.2 ZK-SNARKs in Zcash

Protocolli di non interactive zero-knowledge proof per affermazioni generali non sono possibili senza l'utilizzo di una CRS (common reference string). Essa deve essere nota sia al prover che al verifier. Generalmente si assume che la CRS sia generata onestamente. In pratica essa viene prodotta da una terza parte verificata oppure si utilizza un *multi-party computational protocol*.

Come descritto in [14], l'acronimo SNARKS (*succint non-interactive arguments of knowledge*) indica:

- **Succint**: la grandezza dei messaggi è ridotta;
- **Non-interactive**: non ci sono interazioni o sono minime; Solitamente è presente un fase di setup e dopo un messaggio dal prover al verifier. La proprietà *public verifier* significa che chiunque può verificare la prova senza interagire nuovamente;
- **ARguments**: per generare gli argomenti si ritengono valide le assunzioni di difficoltà computazionale fatte in modo da ottenere la proprietà di *computational soundness*;
- **of Knowledge**: non è possibile per il prover costruire la prova/argomento senza conoscere il cosiddetto witness.

Aggiungendo il prefisso zero knowledge, si aggiunge la proprietà che durante l'interazione il verifier non conosce nulla ad eccezione della validità dell'affermazione.

Le ZK-SNARKs attualmente implementate si compongono di quattro elementi:

1. **problema polinomiale:** il programma che deve essere controllato viene convertito in un problema di calcolo di un'equazione quadratica polinomiale;
2. **succinctness:** anzichè verificare tutta la funzione, il verifier sceglie un punto segreto s ;
3. **cifratura omomorfica:** viene utilizzata una funzione di cifratura E che possiede alcune proprietà omomorfe;
4. **zero knowledge:** vengono nascosti i valori grazie alla moltiplicazione per un valore segreto. Il verifier può controllare la correttezza della prova senza conoscere i valori cifrati.

Solitamente le ZK-SNARKs producono prove di lunghezza costante per qualsiasi affermazione e necessitano di un tempo di verifica ridotto. D'altra parte richiedono protocolli per la generazione di CRS molto lunghi e computazionalmente dispendiosi per essere generati in maniera distribuita [12].

3.2.1 Quadratic span program

Secondo la pubblicazione “Quadratic Span Programs and Succinct NIZKs without PCs” [15], il problema QSP (Quadratic Span Program) è particolarmente adatto alle ZK-SNARKs. QSP è formato da un insieme di polinomi e l'obiettivo è quello di trovare una combinazione lineare dei polinomi che sia multiplo di un altro polinomio dato. Quindi QSP su un campo F per input di lunghezza n è formato da:

- polinomi $v_0, \dots, v_m, w_0, \dots, w_m$ su un campo F ;
- un polinomio obiettivo t su campo F ;
- una funzione iniettiva $f : \{(i, j) | 1 \leq i \leq n, j \in \{0, 1\}\} \rightarrow \{1, \dots, m\}$.

Un input u è accettato da QSP se e solo se esistono tuple $a = (a_1, \dots, a_m)$ e $b = (b_1, \dots, b_m)$ appartenenti da F tale che:

- $a_k, b_k = 1$ se $k = f(i, u[i])$ per alcuni i (dove $u[i]$ è l' i -esimo bit di u);
- $a_k, b_k = 0$ se $k = f(i, 1 - u[i])$ per alcuni i ;
- il polinomio obiettivo t divide $v_a w_b$ dove $v_a = v_0 + a_1 v_1 + \dots + a_m v_m$ e $w_b = w_0 + b_1 w_1 + \dots + b_m w_m$.

Si noti che si è liberi di scegliere le tuple a e b se $2n$ è minore di m . Questo significa che QSP ha senso solo per input di una certa lunghezza. Come analogia di soddisfacibilità di una formula booleana, i valori a_1, \dots, a_m e b_1, \dots, b_m sono assegnamenti di variabili, o in generale, testimoni per un problema in NP. Per la dimostrazione di come QSP possa

essere ridotto ad un problema NP si veda [15].

Un aspetto importante è come verificare in maniera efficiente QSP. La verifica consiste nel controllare se un polinomio divide un altro polinomio. Questo può essere facilitato se il prover fornisce un altro polinomio h tale che $th = v_a w_b$. Moltiplicare due polinomi molto grandi rimane comunque un “compito difficile”. Al posto di calcolare $v_a w_b$, il verifier sceglie un punto casuale s (una parte del *toxic waste* di Zcash), calcola $t(s)$, $v_k(s)$ e $w_k(s)$ per ogni k , e infine ottiene $v_a(s)$ e $w_b(s)$. Per verificare il risultato del problema, $t(s)h(s) = v_a(s)w_b(s)$. Controllare l'identità di un polinomio per un punto singolo, rispetto a controllare tutti i punti, riduce la sicurezza. Comunque la probabilità che il prover possa barare è molto piccola, dato che non conosce s e il numero di soluzioni, considerati i gradi dei polinomi, sono minuscole rispetto ai possibili valori del punto s (pari al numero di elementi di un campo). Quindi la verifica diventa sicura nella pratica.

3.2.2 ZK-SNARKs nel dettaglio

In Zcash, il circuito (transazione del verifier) è unico e perciò i polinomi di QSP sono sempre gli stessi. Questo permette che la fase di setup venga effettuata una sola volta e riutilizzata per tutte le transazioni, di cui cambia solamente l'input u . Durante la fase di setup, quando viene generata la common reference string, il verifier sceglie a caso un valore segreto s sul campo F e calcola il valore dei polinomi per quel dato punto. Inoltre, usa una funzione di cifratura specifica E e pubblica $E(v_k(s))$ e $E(w_k(s))$ nel CRS. Il CRS contiene molti altri valori che rendono la verifica più efficiente e garantiscono la proprietà di zero-knowledge. La cifratura E utilizzata possiede alcune proprietà omomorfe, che permettono di calcolare $E(v(s))$ senza conoscere $w_k(s)$.

Si consideri un esempio semplificato, in cui si verifica un polinomio cifrato per un punto segreto s . Si scelga un campo F , solitamente una curva ellittica, e un generatore g . La cifratura è semplicemente $E(x) = g^x$.

All'inizio il verifier sceglie un valore s in F e pubblica la CRS contenente $E(s^0), E(s^1), E(s^2), \dots, E(s^d)$, dove d è il grado massimo del polinomio f . Ora bisogna “dimenticare” s . In Zcash s fa parte del *toxic waste*, poiché qualunque prover conosca s e i successivi segreti è in grado di generare prove false, calcolando gli zeri del polinomio.

Usando questi valori il prover calcola $E(f(s))$. Ad esempio:

$$\begin{aligned} f(x) = 4x^2 + 2x + 4 &\rightarrow E(f(s)) = E(4x^2 + 2x + 4) \\ &= g^{4s^2 + 2s + 4} \\ &= E(s^2)^4 E(s^1)^2 E(s^0)^4 \end{aligned} \tag{1}$$

Quindi il prover può calcolare il valore cifrato del polinomio senza conoscere s . Per verificare che il prover cifri correttamente il polinomio (ricordandoci che s è stato distrutto),

il verifier sceglie α e pubblica $E(\alpha s^0), E(\alpha s^1), E(\alpha s^2), \dots, E(\alpha s^d)$. Anche α viene distrutto.

Allo stesso modo di come è stato mostrato prima, il prover calcola $E(\alpha f(s))$. Infine pubblica $A = E(f(s))$ e $B = E(\alpha f(s))$.

A questo punto, grazie ad una *pairing function* e il verifier può controllare che A e B siano corretti. La pairing function e la curva ellittica devono essere scelte insieme, in modo che per ogni x e y valga la proprietà:

$$e(g^x, g^y) = e(g, g)^{xy}$$

Usando la pairing function, il verifier verifica che $e(A, g^\alpha) = e(B, g)$. Si noti che il verifier conosce g^α poichè è contenuto nella CRS. Quindi il verifier verifica queste uguaglianze:

$$\begin{aligned} e(A, g^\alpha) &= e(g^{f(s)}, g^\alpha) = e(g, g)^{f(s)\alpha} \\ e(B, g) &= e(g^{f(s)}, g) = e(g, g)^{f(s)} \end{aligned} \tag{2}$$

Un prover malevolo non dovrebbe essere in grado di generare dei valori A e B che soddisfino la pairing function.

Per aggiungere la proprietà di zero-knowledge, il prover sceglie a caso un valore δ , e anzichè inviare A e B invia

$$A' = E(\delta + f(s)) = E(\delta)A \quad B' = E((\delta + f(s))) = E(\alpha)^\delta B$$

3.2.3 ZK-SNARKs per il problema QSP

Si ricordi che nel problema QPS vengono dati i polinomi $v_0, \dots, v_m, w_0, \dots, w_m$ e un polinomio obiettivo t di grado al massimo d e una stringa binaria u . Il prover calcola i valori $a_1, \dots, a_m, b_1, \dots, b_m$ e un polinomio h tale che:

$$th = (v_0 + a_1v_1 + \dots + a_mv_m)(w_0 + b_1w_1 + \dots + b_mw_m)$$

A differenza dell'esempio precedente il problema QSP prevede un insieme di polinomi fissati. Quindi vengono pubblicati i valori dei seguenti polinomi:

- $E(\gamma), E(\beta_v\gamma), E(\beta_w\gamma)$
- $E(\beta_vv_1(s)), \dots, E(\beta_vv_m(s))$
- $E(\beta_wv_1(s)), \dots, E(\beta_wv_m(s))$
- $E(\beta_vt(s)), E(\beta_wt(s))$

Il prover utilizza la riduzione mostrata precedentemente per trovare il polinomio h e i valori $a_1 \dots a_m$ e $b_1 \dots b_m$.

Ricordando che la funzione iniettiva f limita i valori $a_1 + \dots + a_m$ e $b_1 + \dots + b_m$, fino

a quando m è molto grande, ci sono numeri che, indipendentemente dall'input scelto, non compaiono come output di f . Questi indici non limitati e vengono chiamati I_{free} . Si definisce

$$v_{free}(x) = \sum_k a_k v_k(x)$$

dove k appartiene a tutto l'intervallo di I_{free} . Per $w(x) = b_1 w_1(x) + \dots + b_m w_m(x)$ la prova consiste in:

- $V_{free} = E(v_{free}(s)), \quad W_{free} = E(w(s)), \quad H = E(h(s))$
- $V'_{free} = E(\alpha v_{free}(s)), \quad W'_{free} = E(\alpha w(s)), \quad H' = E(\alpha h(s))$
- $Y = E(\beta_v v_{free}(s) + \beta_w w(s))$

dove l'ultimo punto controlla che siano utilizzati i polinomi corretti.

Il verifier controlla le seguenti uguaglianze con la funzione di pairing:

1. $e(V'_{free}, g) = e(V_{free}, g^\alpha) \quad e(W', E(1)) = e(W, E(\alpha)) \quad e(H', E(1)) = e(H, E(\alpha))$
2. $e(E(\gamma), Y) = e(E(\beta_v \gamma), V_{free}) e(E(\beta_w \gamma), W)$
3. $e(E(v_0(s)) E(v_{in}(s) V_{free}, E(w_0(s)) W) = e(H, E(t(s)))$

I valori di a_k dove k non è un free index possono essere calcolati direttamente dall'input u , il verifier calcola la parte mancante della somma completa di v :

$$E(v_{in}(s)) = E\left(\sum_k a_k v_k(s)\right)$$

dove k sono indici non in I_{free}

Bisogna comprendere che la pairing function permette solo alcune computazioni limitate su valori cifrati. Possiamo fare un numero di addizioni a piacere ma solo una singola moltiplicazione.

Il punto 2 testimonia la prova. Quindi il verifier per verificarla esegue:

$$\begin{aligned}
 e(E(\gamma), Y) &= e(E(\gamma), E(\beta_v v_{free}(s) + \beta_w w(s))) \\
 &= e(g, g)^{\gamma(\beta_v v_{free}(s) + \beta_w w(s))} \\
 &= e(g, g)^{(\beta_v \gamma) v_{free}(s)} e(g, g)^{(\beta_w \gamma) w(s)} \\
 &= e(E(\beta_v \gamma), e(V_{free}(s))) e(E(\beta_w \gamma), E(w(s))) \\
 &= e(E(\beta_v \gamma), V_{free}) e(E(\beta_w \gamma), W)
 \end{aligned} \tag{3}$$

L'aggiunta delle proprietà di zero-knowledge consiste nel fatto che il prover trasli alcuni valori di una quantità casuale segreta e bilanci lo spostamento nell'altra parte dell'equazione. Il prover sceglie in maniera casuale δ_{free} e δ_w ed esegue questa sostituzione:

- $v_{free}(s)$ viene sostituito da $v_{free}(s) + \delta_{free}t(s)$
- $w(s)$ viene sostituito da $w(s) + \delta_w t(s)$

Grazie a queste sostituzioni i valori V_{free} e W , che contengono i fattori di witness codificati, diventano indistinguibili da valori casuali e quindi diventa impossibile estrarre i witness.

3.2.4 Zcash ceremony

Come è già stato detto, le ZK-SNARKs richiedono una fase di setup in cui una CRS viene generata con una certa struttura. La common reference string viene anche chiamata parametri pubblici del sistema, e viene utilizzata per costruire le verifying proof [16]. Questa fase di setup è nota come *ceremony*. Durante la ceremony vengono utilizzati dei parametri, definiti toxic waste, che devono essere cancellati, altrimenti chiunque entri in possesso di queste informazioni potrebbe produrre pseudo prove fraudolente. Questa cerimonia richiede un quantitativo di tempo, energia e potenza computazionale non indifferente per assicurarsi che tutto venga svolto in sicurezza.

Per aumentare la sicurezza, Zcash ha utilizzato un protocollo multi-party per generare la common reference string. Fino a quando almeno uno dei partecipanti non è malevolo, nessuno può costruire prove fraudolente ad eccezione di una piccola probabilità. In pratica è necessaria la collusione di tutti i partecipanti per riprodurre il toxic waste. Comunque il protocollo fornisce una forte zero-knowledge, garantita anche nel caso in cui tutti i partecipanti siano maliziosi.

In Zcash l'ammontare di una transazione privata viene verificata dalle ZK-SNARKs. Quindi chiunque venga a conoscenza del toxic waste può costruire e verificare ZK-SNARKs e di conseguenza generare una quantità illimitata di moneta potenzialmente senza che nessuno se ne accorga.

Le persone scelte per la ceremony devono essere geograficamente separate e non devono essere note fino al completamento del protocollo. Sono stati utilizzati dei computer nuovi, mai collegati ad alcuna rete. Prima di essere accesi, le schede bluetooth e wifi sono state fisicamente rimosse. Il tutto è stato documentato da audio e video alla presenza di giornalisti [17].

La “genesis Ceremony” è stata organizzata nell'ottobre del 2016 e hanno partecipato Zooko Wilcox, fondatore di Zcash, e cinque altre persone considerate eticamente corrette e con la giusta preparazione nell'ambito della sicurezza informatica. Cinque persone sono note mentre il sesto partecipante è anonimo. In occasione del *sampling fork* del 2018 è stata organizzata una seconda ceremony, chiamata “Powers of Tau” ceremony. Questa volta hanno partecipato circa 90 tra persone e organizzazioni per rendere la creazione della CRS ancora più sicura contro la collusione. La seconda ceremony è stata resa necessaria dalla scoperta di una vulnerabilità presente durante la genesis

Ceremony.

Nel 2018 Ariel Gabizon, un crittografo allora appartenente alla compagnia di Zcash, ha trovato una vulnerabilità nell'implementazione di una libreria utilizzata per generare i parametri pubblici delle ZK-SNARKs [18]. Questa vulnerabilità è stata risolta nell'aggiornamento di ottobre 2018, Sapling, rigenerando dei nuovi parametri. Secondo quanto esposto, partendo dai transcript generati nell'MPC setup, era possibile ottenere alcuni parametri facenti parte del toxic waste. Con queste informazioni era possibile creare delle false prove riguardo gli *shielded address*, e quindi generare ZEC dal nulla. Per verificare che la vulnerabilità non fosse stata utilizzata, la Zcash foundation ha disposto che per spostare ZEC contenuti in uno shielded address dal fork Sprout a quello Sapling la transazione doveva essere visibile. In questo modo, durante la migrazione degli indirizzi, la Zcash foundation ha potuto osservare che non fosse stato trasferito un totale di ZEC maggiore rispetto a quello creato. Questo però non prova che la vulnerabilità non sia stata sfruttata. Infatti una certa quantità di ZEC potrebbe essere bloccata perchè gli utenti hanno perso le loro chiavi e contemporaneamente degli attaccanti potrebbero aver creato quella stessa quantità o meno senza che nessuno se ne sia accorto. Inoltre c'è la possibilità che gli ZEC che non sono stati ancora trasferiti in shielded-address di Sapling vengano bloccati, qualora il saldo di ZEC creati meno ZEC trasferiti diventi negativo.

3.3 Bulletproof in Monero

Monero utilizza diversi meccanismi per garantire l'anonimato. Essi sono descritti nel paragrafo 5.1. Per nascondere l'importo delle transazioni, Monero cifra l'ammontare di ogni transazione in modo che solo il destinatario possa decifrarla. Per garantire che il valore degli output di una transazione sia uguale alla somma degli input e delle fee, Monero utilizza una *zero-knowledge range proof*, chiamata *Bulletproof*.

3.3.1 Prerequisiti

Uno schema di *commitment* è una primitiva crittografica che permette a un attore di impegnarsi riguardo ad un valore o un'affermazione e tenerla nascosta agli altri con la possibilità di rivelare il valore scelto successivamente. Gli schemi di commitment sono progettati affinché l'attore non possa cambiare il valore una volta effettuato il commitment.

Riportiamo ora alcune definizioni espresse nel paper di Bottle et al. [12] necessarie per comprendere le zero knowledge proof in generale e più nello specifico la Bulletproof utilizzata in Monero.

Un avversario A che esegue algoritmi in PPT (probabilistic polynomial time) è una macchina di Turing probabilistica interattiva che computa in tempo polinomiale rispetto al

parametro di sicurezza λ .

Definizione 5 (Discrete log relation)

Per tutti gli avversari A che eseguono algoritmi in PPT e per ogni intero $n \geq 2$ esiste una funzione trascurabile $\mu(\lambda)$

$$P \left[G = \text{Setup}(1^\lambda), g_1, \dots, g_n \leftarrow G \right. \\ \left. : \exists a_1 \neq 0 \wedge \prod_{i=1}^n g_i^{a_i} = 1 \right] \leq \mu(\lambda) \quad (4)$$

Si afferma che $\prod_{i=1}^n g_i^{a_i} = 1$ è una relazione del logaritmo discreto non banale per g_1, \dots, g_n . La relazione del logaritmo discreto afferma che un avversario A non può trovare una relazione non banale tra elementi del gruppo scelti a caso. Per $n \geq 1$ l'assunzione è equivalente all'assunzione del logaritmo discreto.

Definizione 6 (Commitment)

Uno schema di commitment non interattivo è formato da una coppia di algoritmi PPT (Setup, Com). L'algoritmo di setup $pp \leftarrow \text{Setup}(1^\lambda)$ genera i parametri pubblici pp per lo schema, con il parametro di sicurezza λ . L'algoritmo Com_{pp} definisce una funzione $M_{pp} \times R_{pp} \rightarrow C_{pp}$, dove pp determina lo spazio del messaggio M_{pp} , l'insieme dei numeri casuali R_{pp} e lo spazio di commitment C_{pp} . Per un messaggio $x \in M_{pp}$, l'algoritmo genera un $r \leftarrow R_{pp}$ scelto a caso in maniera uniforme, e computa il commitment $com = \text{Com}_{pp}(x; r)$. Per semplicità si scriverà $com = \text{Com}_{pp}$.

Definizione 7 (Homomorphic commitment)

Uno schema di commitment omomorfo è uno schema di commitment non interattivo tale che M_{pp}, R_{pp}, C_{pp} sono tutti gruppi abeliani e per ogni $x_1, x_2 \in M_{pp}$ e $r_1, r_2 \in R_{pp}$ avviene che:

$$\text{Com}(x_1; r_1) + \text{Com}(x_2; r_2) = \text{Com}(x_1 + x_2; r_1 + r_2)$$

Definizione 8 (Hiding commitment)

Uno schema di commitment si definisce nascosto se per ogni avversario A che esegue algoritmi PPT esiste una funzione $\mu(\lambda)$ tale che:

$$\left| P \left[b = b' \mid \begin{array}{l} pp \leftarrow \text{Setup}(1^\lambda) \\ (x_0, x_1) \in M_{pp}^2 \leftarrow A(pp), b \leftarrow \{0, 1\}, r \leftarrow R_{pp}, \\ com = \text{Com}(x_b, r), b' \leftarrow A(pp, com) \end{array} \right] - \frac{1}{2} \right| \leq \mu(\lambda) \quad (5)$$

cioè la probabilità che una PTM calcoli un b' uguale a b che risolva l'hiding commitment deve essere minore a una funzione trascurabile $\mu(\lambda)$. In maniera informale significa che un attaccante A , a meno di una funzione trascurabile $\mu(\lambda)$, non è in grado di calcolare il valore nascosto x_b . Se $\mu(\lambda) = 0$ allora lo schema si definisce *perfectly hiding*. Questo significa che un avversario con un'illimitata capacità computazionale, anche provando per un tempo infinito non sarà mai in grado di calcolare il valore segreto x_b .

Definizione 9 (Binding commitment)

Uno schema di commitment si definisce *binding* se per ogni avversario A che esegue algoritmi PPT esiste una funzione trascurabile $\mu(\lambda)$ tale che:

$$P \left[Com(x_0; r_0) = Com(x_1; r_1) \wedge x_0 \neq x_1 \mid \begin{array}{l} pp \leftarrow Setup(1^\lambda) \\ x_0, x_1, r_0, r_1 \leftarrow A(pp) \end{array} \right] \leq \mu(\lambda) \quad (6)$$

dove la probabilità termina nella fase di Setup. Binding commitment significa che a meno di una funzione trascurabile $\mu(\lambda)$, l'attaccante A riesce a collegare due commitment con messaggi e numeri casuali diversi. Ovvero una volta data la stringa b , non è possibile trovare un'altro commitment diverso da quello che ha generato b , a meno di una funzione trascurabile. Se $\mu(\lambda) = 0$ allora lo schema si dice *perfectly binding*.

È impossibile progettare un commitment scheme che sia *perfectly binding* e *perfectly hiding*. Quindi quando si progetta un commitment scheme e un proof system bisogna stabilire quale sia la proprietà più importante.

Definizione 10 (Pedersen commitment)

Un Pedersen Commitment è uno schema di commitment in cui $M_{pp}, R_{pp} = Z_p$ e $C_{pp} = G$ di ordine p . Quindi: $Setup : g, h \leftarrow G$ e $Com : (x, r) = (g^x, h^r)$

Definizione 11 (Pedersen vector commitment)

Un Pedersen vector commitment è un Pedersen commitment in cui il valore x è rappresentato da una stringa. In particolare $M_{pp} = Z_p^n, R_{pp} = Z_p$ e $C_{pp} = G$ di ordine p . Quindi: $Setup : g = (g_1, g_2, \dots, g_n), h \leftarrow G$ e $Com : (x = (x_1, x_2, \dots, x_n), r) = h^r, g^x = h^r \prod_i g_i^{x_i} \in G$

Il pedersen vector commitment è *perfectly hiding* e *computationally binding* sotto le assunzioni del logaritmo discreto, ovvero per violare la proprietà di binding bisogna risolvere il problema del logaritmo discreto. Se si imposta $r = 0$, il commitment è *binding* ma non *hiding*.

Definizione 12 (Public coin)

Un argomento di knowledge $(Setup, P, V)$ viene chiamato *public coin* se tutti i messaggi inviati dal verifier al prover sono scelti in maniera casuale e sono indipendenti dai messaggi del prover.

Definizione 13 (Perfect special honest-verifier zero-knowledge)

Perfect special honest-verifier zero-knowledge (SHVZK) è un argomento di knowledge in cui la sfida del verifier di verificare i valori può essere simulata efficientemente senza conoscere il witness. Formalmente, se esiste un simulatore S tale che per ogni coppia di "avversari" interattivi A_1, A_2 :

$$P \left[(\sigma, u, w) \in R \wedge A_1(tr) = 1 \mid \begin{array}{l} \sigma \leftarrow Setup(1^\lambda), (u, w, \rho) \leftarrow A_2(\sigma), \\ tr \leftarrow \langle P(\sigma, u, w), V(\sigma, u, \rho) \rangle \end{array} \right] \quad (7)$$

$$= P \left[(\sigma, u, w) \in R \wedge A_1(tr) = 1 \mid \begin{array}{l} \sigma \leftarrow Setup(1^\lambda), (u, w, \rho) \leftarrow A_2(\sigma), \\ tr \leftarrow S(u, \rho) \end{array} \right]$$

dove ρ è il public coin casuale utilizzato dal verifier e $(tr) = 1$ è una trascrizione considerata valida.

In questa definizione “l’avversario” sceglie una distribuzione sull’affermazione e sul testimone, ma non è in grado di distinguere tra le trascrizioni (tr) simulate e quelle generate onestamente da affermazioni e testimoni validi.

Definizione 14 (Range proof)

Una zero-knowledge range proof è un argomento di knowledge SHVZK, con un commitment scheme $(Setup, Com)$ su uno spazio di messaggio M_{pp} , che è un insieme con un ordine totale, e una relazione R_{range} :

$$R_{range} : ((pp), (com, l, r), (x, p)) \in R_{range} \leftrightarrow com = Com(x, p) \wedge l \leq x < r$$

ovvero è una prova zero-knowledge che testimonia che il valore segreto x appartiene ad un intervallo stabilito.

3.3.2 Bulletproof

Bottle et al. [19] hanno introdotto una comunicazione efficiente inner-product argument, cioè una dimostrazione zero-knowledge per la soddisfacibilità di un circuito aritmetico con una complessità nella comunicazione che cresce in maniera logaritmica rispetto alla grandezza del circuito. Questo protocollo viene utilizzato per costruire la Bulletproof. In particolare si è sfruttato il fatto che un Pedersen commitment V è un elemento dello stesso gruppo G , utilizzato per inner product argument.

Sia $V \in G$ un Pedersen commitment per il valore v usando una variabile randomica γ . Il proof system convince il verifier che $v \in [0, 2^{n-1}]$. Più precisamente viene provata la seguente relazione:

$$\{(g, h \in G, V, n; v, \gamma \in Z_p) : V : h^\gamma g^v \wedge v \in [0, 2^{n-1}]\}$$

Per la dimostrazione della Bulletproof si veda il capitolo 4 di [12]. Inoltre, per costruzione, la Bulletproof permette di aggregare diverse range proof in un’unica dimostrazione zero-knowledge. Infine, grazie all’utilizzo dell’euristica di Fiat-Shamir il protocollo è stato reso non interattivo. In particolare anzichè richiedere al verifier la generazione di un numero casuale, si utilizza il risultato di funzione di hash che utilizza come input alcuni parametri noti ad entrambi gli attori.

Bulletproof, come le altre range proof utilizzate nella confidential transaction di Monero, sono computational binding. Quindi un avversario in grado di violare le assunzioni

del logaritmo discreto potrebbe generare range proof per un valore non appartenente all'intervallo. D'altra parte Bulletproof è perfectly-hiding o perfect-zero-knowledge, pertanto nessun attaccante potrà apprendere il valore presente nel commitment.

Per una cryptovaluta la proprietà di binding è più importante rispetto a quella di hiding [20] poiché un attaccante in grado di violare la proprietà di binding di uno schema di commitment o la proprietà di soundness di un proof system, potrebbe generare coin dal nulla. Come riportato dagli autori [12], sembra difficile creare una Bulletproof per un binding commitment. Inoltre per definizione, in un perfect binding commitment scheme la grandezza del commitment deve essere almeno pari alla grandezza del messaggio e la compressione è impossibile. Rispetto alle precedenti range proof adottate da Monero, Bulletproof necessita di un tempo di verifica minore e riduce la grandezza delle transazioni minimizzando lo spazio in maniera efficiente.

L'assunzione del logaritmo discreto viene ritenuta valida per i computer "classici" ma non si hanno certezze contro un attaccante che utilizzi un quantum computer. Per questo motivo gli autori della Bulletproof hanno utilizzato una tecnologia di Ruffing e Malavolta [RM] per assicurarsi che sebbene sia solamente computationally binding, in futuro sarà possibile passare ad un proof system che sia perfectly binding e sicuro anche contro un quantum computer. Le *Succinct quantum secure range proof* rimangono ancora un problema aperto.

3.4 Confronto tra ZK-SNARKs e Bulletproof

Le ZK-SNARKs producono prove di lunghezza costante per qualsiasi affermazione e necessitano di un tempo di verifica ridotto. Inoltre, come indica l'acronimo SNARKs, questo tipo di dimostrazione genera una prova di dimensione ridotta. Lo svantaggio di questo sistema è la necessità di una common reference string. Per generare la CRS è necessario un *trusted setup* da cui dipende la validità delle prove. Al contrario nella Bulletproof non è richiesto alcun trusted setup. Il tempo di verifica è lineare e quindi più lento delle ZK-SNARKs. La dimensione delle prove è piccola ma maggiore di quelle prodotte dalle ZK-SNARKs. Per costruzione, la Bulletproof è comunque efficiente nell'aggregazione delle prove. Entrambe le prove sono perfectly hiding e quindi non sono perfectly binding. Nella tabella 1 viene riassunto il confronto tra i due tipi di zero-knowledge proof.

	SNARKs	Bulletproof
Algorithmic complexity: prover	$O(N * \log(N))$	$O(N * \log(N))$
Algorithmic complexity: verifier	$\sim O(1)$	$O(N)$
Communication complexity (proof size)	$\sim O(1)$	$O(\log(N))$
- size estimate for 1 TX	Tx: 200 bytes, Key: 50 MB	1.5 kb
- size estimate for 10.000 TX	Tx: 200 bytes, Key: 500 GB	2.5 kb
Trusted setup required?	YES	NO
Post-quantum secure	NO	NO
Crypto assumptions	Strong	Discrete log

Tabella 1: Confronto tra ZK-SNARKs e Bulletproof

4 Esempi di voto elettronico

In questo capitolo vengono descritti i principali sistemi di voto che utilizzano una blockchain. Essendo soluzioni proprietarie non sempre la descrizione risulta precisa e dettagliata. Per reperire le informazioni di ogni sistema di voto sono stati utilizzati i whitepaper pubblicati dai creatori ed eventuali articoli scientifici che ne analizzano le proprietà di sicurezza.

Non esiste un sistema di voto elettronico riconosciuto universalmente come valido. Prima di adottare un sistema di voto elettronico bisogna risolvere alcune questioni fondamentali come l'autenticazione e l'identità digitale, la coercizione, l'anonimato dell'elettore e la verificabilità del voto. Votebook [21] e Votewatcher [22] risolvono alcune di queste problematiche riproponendo un voto in presenza, come nelle votazioni su carta, sostituendo le schede elettorali con le EVM. L'utilizzo della blockchain favorisce l'integrità, la verificabilità e la trasparenza del sistema.

Alcuni sistemi propongono votazioni anche da remoto, con le difficoltà e le complicazioni che questo comporta. Alcuni si basano sull'utilizzo della blockchain Ethereum ed in particolare degli smart contract, script la cui esecuzione è decentralizzata. La soluzione più solida e documentata è Polys [23].

Altri sistemi, come Crypto-voting [24] e Agora [25], hanno un'architettura complessa ed integrano più blockchain.

Nel seguito vengono descritti sistemi di voto adottati in elezioni politiche. I-voting è il sistema di voto utilizzato in Estonia. L'Estonia è stato il primo paese al mondo ad effettuare delle elezioni tramite internet. Voatz [26] invece è un sistema di voto americano utilizzato nella convention dei democratici in Massachusetts.

Infine viene descritto VoteCoin [27], un progetto che diversamente dagli altri non utilizza le proprietà della blockchain per costruire un sistema di voto coerente con gli attuali sistemi di voto ma propone un'evoluzione dei sistemi di voto in modo che assomigli maggiormente allo scambio di valore, tipico della blockchain.

4.1 Votebook

Votebook [21] è una piattaforma di voto online nata in seguito alla competizione sponsorizzata dal Kaspersky lab e dal giornale Economist nel 2016 [28]. Votebook partendo dal presupposto che l'autenticazione degli elettori e la sicurezza dei dispositivi possono essere compromesse da remoto, ha deciso di implementare un sistema di voto che ricalca gli attuali sistemi di voto americani, ovvero con seggi, cabine e schede elettorali aggiungendo l'utilizzo della blockchain. L'idea consiste nell'utilizzare una blockchain *permissioned*, in cui ciascuna voting machine agisce come un nodo della rete. I nodi appartenenti alla rete sono autorizzati da un'autorità centrale, da cui ricevono una chiave di cifratura simmetrica.

Prima dell'apertura della votazione, ciascuna EVM genera una coppia di chiavi e invia la chiave pubblica all'autorità centrale. L'autorità centrale raccoglie le diverse chiavi e inoltra ad ogni nodo una tabella contenente gli ID dei nodi e le rispettive chiavi pubbliche.

Durante la votazione, ciascuna EVM raccoglie i voti e li organizza in blocchi. Per evitare collisioni, i nodi propongono il proprio blocco basandosi su un protocollo *time-based*. Ogni blocco è formato dall'ID del nodo, il timestamp, e tre segmenti di validazione:

- un insieme di righe che riportano i voti espressi dagli elettori. Ogni riga è formata da due colonne. La prima colonna contiene un identificativo costruito dalla concatenazione dell'hash del voter ID e dell'hash del ballot ID. La seconda contiene il voto espresso;
- hash del blocco precedente;
- firma digitale dell'hash del blocco, che garantisce l'integrità del blocco proposto da un dato nodo.

Ogni altro nodo verifica l'integrità e la validità del blocco utilizzando la chiave pubblica corrispondente all'ID del nodo che propone il blocco. Se durante una votazione si svolgono più elezioni, viene creata una blockchain per ogni elezione.

Per rilevare il riempimento delle schede elettorali, Votebook implementa un'altra blockchain che registra anonimamente i votanti. I blocchi di questa blockchain contengono una lista di hash degli ID dei votanti, l'hash del blocco precedente e l'hash del blocco cifrato con la chiave privata della voting machine. C'è il rischio che un avversario confronti i blocchi contenenti le identità degli elettori con i blocchi contenenti i voti espressi per effettuare inferenze statistiche sul voto espresso. Quindi, oltre a rimescolare l'ordine degli elettori nel blocco, il registro degli elettori deve essere mantenuto privato e mai pubblicato. Per preservare la privacy, i nodi cifrano gli ID degli elettori con la chiave simmetrica ricevuta dall'autorità centrale [21].

4.2 Votewatcher

Votewatcher [22] è probabilmente uno dei sistemi di voto basato su blockchain più maturo e testato. Costruito a partire dal 2016 dalla blockchain Technologies Corporation (BTC), una grande azienda che fornisce servizi basati su blockchain. Come per Votebook, la user experience dei votanti è la stessa dei sistemi attualmente in uso. Al seggio viene consegnata una scheda elettorale con tre QR code, uno per l'indirizzo della blockchain, uno rappresentante l'ID della scheda e uno contenente l'ID dell'elezione. Quando viene scansionato il QR code, il voto viene inviato all'indirizzo del candidato su una blockchain locale (offline). Inoltre la macchina memorizza una copia di tutti i bussolotti. Per lo scrutinio, tutti i dati (copia dei bussolotti, blockchain e metadati) vengono salvati su un dispositivo di memorizzazione esterno e poi le diverse EVM vengono collegate online per il conteggio finale. A questo punto è possibile utilizzare il blockchain explorer per ottenere il risultato delle elezioni. Dato che utilizzare Bitcoin è molto costoso, Votewatcher ha creato una propria copia della blockchain di Bitcoin senza fee [28].

4.3 Polys

Polys [23] è una piattaforma per votazioni online sviluppata dal Kaspersky lab, nel 2020. Questa piattaforma permette una soluzione ibrida, ovvero è possibile votare sia tramite dispositivi online che tramite voting machine. Quando un elettore effettua il voto, questo viene firmato con una chiave, dove la chiave è un alias del votante. L'alias è presente nell'elenco del sistema elettorale e assicura che una persona possa effettuare un solo voto. I voti sono cifrati con una *public Election key*. Successivamente il voto viene trasformato in una transazione Ethereum. Ogni transazione viene identificata con un QR code. Il QR code consente all'elettore di controllare, attraverso una speciale applicazione web, che il suo voto sia stato registrato nella blockchain.

Per mantenere la segretezza e il normale svolgimento della votazione, ciascun voto deve essere cifrato, in modo che qualsiasi osservatore non possa dedurre il contenuto del voto. Il voto viene cifrato con una *public Election key*, la cui corrispettiva chiave privata è suddivisa in molteplici parti possedute dai membri del comitato elettorale. Per condividere il segreto si utilizza il *Shamir's Secret Sharing scheme* [29]. Questo segreto condiviso e distribuito assicura una maggiore sicurezza.

Il problema dell'autenticazione dell'utente è presente in qualsiasi sistema di voto remoto. Polys affronta questo problema a seconda del tipo di votazione, ma generalizzando l'elettore deve creare una coppia di chiavi e, dopo essersi autenticato in un qualche sistema gestito dall'organizzatore della votazione, comunicare la propria chiave pubblica. In questo modo l'elettore firma il bussolotto con la propria chiave privata così che sia possibile verificare che il proprio voto è andato a buon fine. Quando il bussolotto viene inviato, grazie all'utilizzo di una zero-knowledge proof, il sistema si assicura che esso

contenga un'opzione valida. Infine per limitare la vendita di voti e la coercizione, si permette all'elettore di votare più volte e durante lo spoglio si terrà conto solo dell'ultimo voto espresso.

Per quanto riguarda il conteggio dei voti, tutti i voti cifrati vengono moltiplicati. Sfruttando la proprietà omomorfica del crittosistema di ElGamal [30], il prodotto dei voti cifrati equivale a cifrare il prodotto dei voti. Poiché ciascun candidato viene identificato da un numero primo, i voti ricevuti da un particolare candidato sono pari al valore dell'esponente del numero che lo rappresenta.

Ottenere i risultati per ciascun candidato non è semplice. Grazie all'utilizzo dell'algoritmo di Shanks [31], si accelera la computazione, sebbene rimanga il quesito sulle performance del sistema in un'elezione su vasta scala. Ottimizzando il processo, con un singolo computer è possibile calcolare in un tempo ragionevole 1,000,000 di voti con 5 possibili scelte. Per favorire la scalabilità, è possibile suddividere i voti da conteggiare, ad esempio per ogni blocco, e poi sommarli insieme [23].

Polys ha deciso di utilizzare la blockchain di Ethereum poiché essa implementa gli smart contract, ovvero script la cui esecuzione è decentralizzata. Con esecuzione decentralizzata si intende che la correttezza dell'esecuzione viene verificata da tutti i partecipanti della rete e che lo script non può essere modificato poiché è stato deployato sulla blockchain. Inoltre ogni eventuale modifica da parte del proprietario viene registrata ed è visibile a tutti i nodi.

Il protocollo di voto descritto da Polys è stato implementato tramite smart contract. In questo modo tutte le diverse fasi vengono svolte in maniera decentralizzata e il procedimento per arrivare ad ottenere i risultati è trasparente agli osservatori.

4.4 Crypto-Voting

Crypto-Voting [24] è l'innovativo sistema di voto elettronico integrato, basato su tecnologia blockchain, finanziato da Sardegna Ricerche nell'ambito del POR FESR Sardegna 2014-2020. Il progetto vuole implementare l'idea dirompente di Crypto-Voting tramite l'utilizzo di due blockchain concatenate, del tipo *one-way pegged sidechain*: una per registrare gli elettori aventi diritto e registrare l'operazione di votazione degli elettori votanti ed un'altra per conteggiare i voti assegnati ai vari candidati. Allo stesso tempo, Crypto-Voting è stata progettata per essere multicanale, versatile e totalmente cloud, rispettando le ultime direttive in materia di privacy. Inoltre, il sistema prevede meccanismi di anti-intrusione e anti-manomissione, la gestione automatica delle liste, l'analisi comparata dei voti provenienti da altri sistemi e la massima sicurezza e trasparenza sull'intero processo, dalla registrazione dei votanti al conteggio dei voti [24].

Il sistema alla base del progetto di Crypto-Voting è proprio quello di due blockchain concatenate, una blockchain privata e una sidechain. La presenza di queste due block-

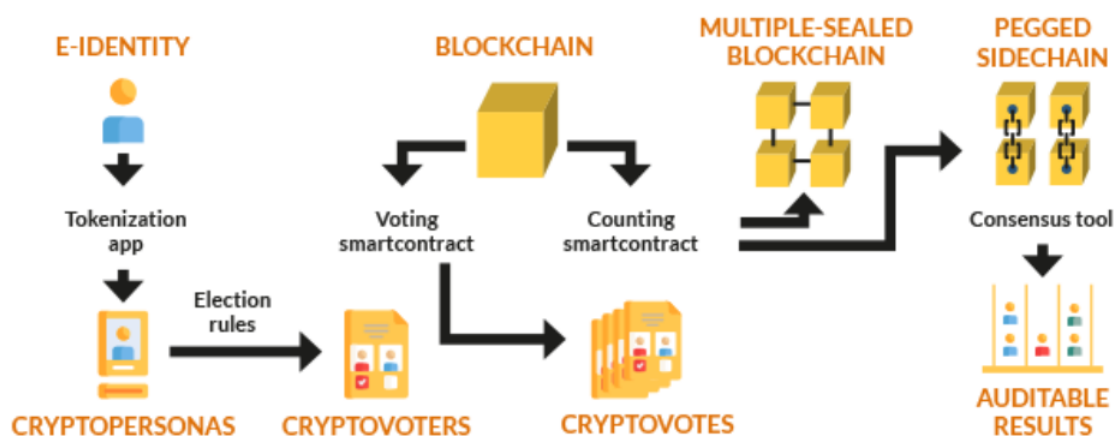


Figura 6: Crypto-voting system

chain separate permette di dividere le operazioni di gestione e conteggio dei voti, aumentando ovviamente, oltre la sicurezza, anche la scalabilità dell'applicazione. Grazie a questo sistema quindi, è possibile effettuare le operazioni comuni a tutte le votazioni, che siano per referendum o elezioni politiche, in totale sicurezza. Il voto è anonimo, in quanto l'unico indizio di chi lo esprime è la chiave generata. È chiaro che nel caso in cui si tratti ad esempio di una raccolta di firme, e quindi di votazioni che per loro natura non siano anonime, è possibile risalire al votante. La caratteristica davvero innovativa di questo sistema di voto elettronico risiede proprio nelle due blockchain concatenate. Negli altri sistemi di voto elettronico già esistenti, viene sfruttata una sola blockchain, che implica il conteggio dei voti nel momento stesso in cui i voti vengono espressi. A differenza degli altri sistemi, questo, divide le operazioni in tre parti:

- preparazione del sistema di votazione;
- gestione del voto e delle espressioni di voto che vengono fatte;
- conteggio dei voti.

Questa divisione permette di incrementare l'efficienza delle operazioni, e soprattutto di non scrivere tutto su una sola blockchain, mettendo a rischio la sicurezza del sistema.

4.5 Agora

Agora [25] è una fondazione svizzera che si occupa di votazioni elettroniche. A partire dal 2015, il team ha sviluppato un'architettura multi-layer che si integra con la blockchain di Bitcoin [4]. Il primo layer viene chiamato Bulletin Board, una DLT (distributed ledger technology) basata sull'architettura Skipchain [32]. Il secondo layer, Cotena,

collega Bulletin Board con Bitcoin, in modo da garantire l'immutabilità e la decentralizzazione dei dati. Il terzo layer, chiamato Valeda, è una rete composta da nodi fidati che validano i risultati dell'elezione. Infine, l'ultimo strato applicativo è Votapp, che rende facile l'interazione con la Bulletin Board.

4.5.1 Bulletin Board

Bulletin Board viene definita una permissioned blockchain, formata da due tipi di nodi:

- *consensus node*: nodi che riconoscono i testimoni di terze parti e hanno il permesso di scrivere sulla DLT;
- *citizen auditor node*: nodi che hanno la sola possibilità di leggere e possono essere gestiti da chiunque nel mondo.

In pratica, Bulletin Board è un database distribuito non modificabile, dove chi ha il permesso può firmare messaggi e statement. L'architettura Skipchain fornisce un meccanismo proattivo di consenso con un alto throughput e un'efficiente validazione delle transazioni [32]. Inoltre Skipchain permette all'applicazione client di navigare arbitrariamente avanti e indietro lungo la blockchain senza la necessità di registrare l'intera blockchain. Questo grazie alla *proof of transaction validity*. Skipchain è ispirata alla *skip list* [33].

Cothority è un insieme di consensus node che collettivamente confermano le transazioni. Questi nodi si occupano di:

- mantenere una copia locale della Bulletin Board;
- ricevere bussolotti cifrati dai votanti e verificare che essi provengano da un elettore autorizzato;
- confermare i blocchi;
- decifrare i bussolotti anonimizzati quando l'elezione è finita, creando bussolotti in chiaro;
- mantenere una copia della Cotena log e monitorare la sua correttezza.

Dall'insieme dei consensus node, un nodo scelto in maniera casuale viene eletto nodo oracolo. Esso si occupa di:

- aggiungere il file di configurazione al Bulletin Board;
- creare un blocco con bussolotti autenticati;
- aggiungere i blocchi confermati al Cotena log e collegarli alla blockchain di Bitcoin.

4.5.2 Cotena

Cotena è un meccanismo di registrazione antimanomissione, costruito sulla blockchain di Bitcoin. Nel layer Cotena, i nodi appartenenti alla Cothority gestiscono un file di log non modificabile, formato da una catena di transazioni Bitcoin. Il file viene costruito grazie all'utilizzo di `OP_RETURN` (uno specifico comando del linguaggio di scripting di Bitcoin), dove viene indicato l'hash dell'ultimo Skipblock. Grazie alle scelte di design, gli utenti che eseguono il software di Agora devono scaricare solo i block header di Bitcoin e i piccoli Merkle tree sotto questi header. Infatti Cotena è stato progettato per evitare di scaricare l'intera blockchain di Bitcoin. Modificare Cotena log vuol dire effettuare un'operazione di doppia spesa su Bitcoin.

Cotena log è una lista di Bulletin Board snapshot catturate periodicamente. Una copia di ogni log è salvata sia dai Cothority node che sulla blockchain di Bitcoin. Per creare una Cotena log, i nodi appartenenti a Cothority generano un nuovo indirizzo Bitcoin, firmano e spediscono una transazione “genesi” Cotena sulla rete Bitcoin. Questa transazione include come statement le chiavi pubbliche dei Cothority node e un hash del primo blocco della Skipblock, oltre ad una somma Bitcoin. Per estendere il Cotena log, Cothority genera un nuovo indirizzo e produce una transazione Bitcoin dall'indirizzo precedente al nuovo. Questa transazione trasferisce la somma della transazione precedente (eccetto le fee che vengono spese) e pubblica come statement lo snapshot dell'ultimo skipBlock inviato dal nodo oracolo. Il Cotena log può essere esteso fino a quando ci sono fondi sufficienti. Qualora il saldo dell'indirizzo non fosse sufficiente per un'altra transazione, vengono aggiunti dei fondi extra da un altro indirizzo.

4.5.3 Valeda

Il layer Valeda serve a fornire la prova finale che Cothority ha mantenuto l'autenticità della Bulletin Board e che quindi i risultati dell'elezione sono validi. La rete Valeda è composta da *citizen auditor node* il cui software, al termine delle elezioni, produce prove crittografiche per verificare diversi processi come la registrazione dei bussolotti, l'anonimizzazione, la decifratura . . .

I citizen auditor node possono essere eseguiti solo da persone o organizzazioni che hanno firmato un contratto con Agora e condotto un KYC (know your customer). I citizen auditor node vengono ricompensati con VOTE token che possono essere scambiati con denaro in un *secondary market*. Prima di ogni elezione, l'organizzazione che intende svolgere l'elezione stanziava una parte del budget proprio per questo fine.

4.5.4 Casting

Prima di iniziare una votazione, l'amministratore crea un file di configurazione dove vengono specificati il tipo di elezione, la data di inizio e fine, la lista degli elettori, la lista dei candidati e altre informazioni. A questo punto tramite una funzione di hash

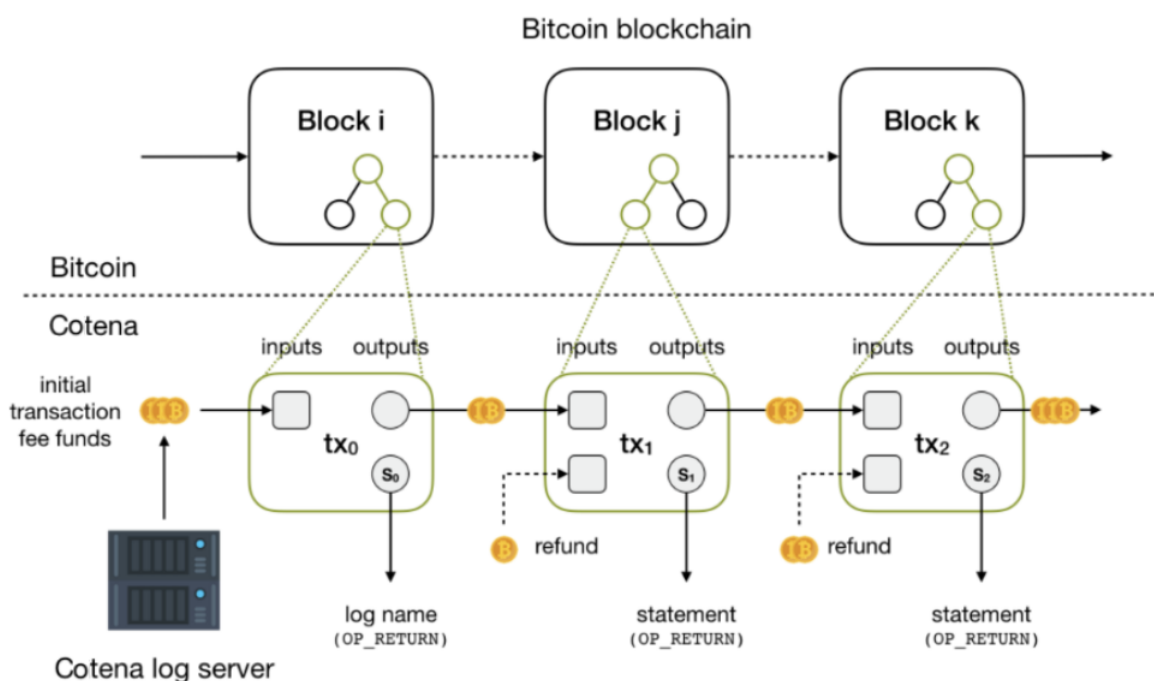


Figura 7: Interazione tra il layer Cotena e Bitcoin

viene generato un ID che identifica l'elezione. Il file di configurazione viene prima firmato dall'organizzatore e poi memorizzato nella Bulletin Board.

Un elettore può effettuare il voto tramite il proprio dispositivo o una voting machine. Quando l'elettore spedisce il bussolotto, esso viene cifrato dal software con la *collective public key* dei Cothority. Il software utilizza il crittosistema di ElGamal. Per verificare che il bussolotto cifrato riflette ancora la scelta dell'elettore, il software permette all'elettore di effettuare un processo noto come *cast as intended validation*. Agora implementa il *cast or challenge validation*, introdotto da Benaloh nel 2006 [34]. In questo metodo un elettore può verificare la correttezza della cifratura tramite una device separata, chiamato voting assistant [35]. Quest'ultimo, partendo da un bussolotto cifrato rivela il corrispettivo testo in chiaro. Se il bussolotto cifrato risulta corretto, l'elettore può spedirlo collegandolo alla propria identità.

Dato che ogni elezione deve garantire la privacy dei votanti, nel network di Agora tutti i bussolotti devono effettuare un processo di anonimizzazione tramite un *mixing network* [36]. Un mixing network è un insieme di server che in sequenza cifra un dato dataset, molteplici volte, dove la correttezza delle cifratura viene testimoniata da una zero-knowledge proof. È sufficiente che uno solo dei nodi sia onesto per garantire la correttezza del protocollo. In particolare, Agora intende implementare il *Neff-shuffle-based mixing network* [36].

I Cothority node controllano la zero knowledge proof della fase di anonimizzazione e, se

è corretta, insieme decifrano i bussolotti. Ciascun Cothority node decifra parzialmente ogni voto anonimo e produce una zero knowledge proof per attestarne la correttezza. I bussolotti in chiaro vengono inviati alla Bulletin Board.

4.6 I-voting

L'Estonia è un paese fortemente votato alla digitalizzazione. Tutti i servizi del cittadino quali scuola, tasse, medicinali, trattamenti medici, legali e amministrativi sono stati digitalizzati. Nel 2005 l'Estonia è stata il primo paese al mondo ad utilizzare un sistema di voto online per le elezioni. Durante le ultime elezioni parlamentari, il 43,8% dei voti è stato effettuato tramite il sistema di e-voting [37]. Secondo il report dei ricercatori dell'università del Michigan, presenti come osservatori durante le elezioni del 2013, nel servizio di e-voting sono presenti diverse vulnerabilità [38].

La carta d'identità è fondamentale per tutti i servizi elettronici. Le carte d'identità estoni sono smartcard con la capacità di effettuare operazioni crittografiche. Queste carte vengono utilizzate per effettuare l'accesso ai diversi servizi digitali e ai servizi di online banking. Nel sistema di I-voting, il votante utilizza la smartcard per autenticarsi e per firmare i bussolotti. Ogni carta contiene due coppie di chiavi RSA, una per l'autenticazione e una per la firma. La smartcard non permette di esportare le chiavi private e quindi tutte le operazioni crittografiche vengono effettuate internamente. Come sicurezza aggiuntiva, a ciascuna chiave è associato un codice PIN, che deve essere inserito per autorizzare ogni operazione. A partire dal 2011, i cittadini estoni possono utilizzare una speciale SIM per smartphone che permette di autenticarsi e di effettuare firme digitali tramite il sistema chiamato Mobile-ID.

Per una maggiore trasparenza del sistema di voto, una parte del codice viene pubblicata su GitHub un mese prima delle elezioni. Inoltre viene registrato un video che documenta l'installazione e la configurazione dei server utilizzati per le elezioni. Malgrado ciò, questi tentativi risultano incompleti e insufficienti a garantire l'integrità del sistema.

Secondo quanto riportato dai ricercatori del Michigan nel 2013 [38], i protocolli di sicurezza non sono ben definiti e, quando presenti, non vengono applicati con la precisione e la rigidità richiesta. Inoltre il sistema si affida eccessivamente alla sicurezza dei client e dei server. Per questo motivo sono possibili attacchi lato client e lato server che possono pregiudicare la correttezza della votazione.

4.6.1 KSI blockchain

Keyless Signature Infrastructure (KSI) [39] è una tecnologia sviluppata dalla compagnia Guardtime a partire dal 2007. Guardtime definisce KSI una *permissioned blockchain* progettata per assicurare l'integrità dei dati e la gestione di asset digitali.

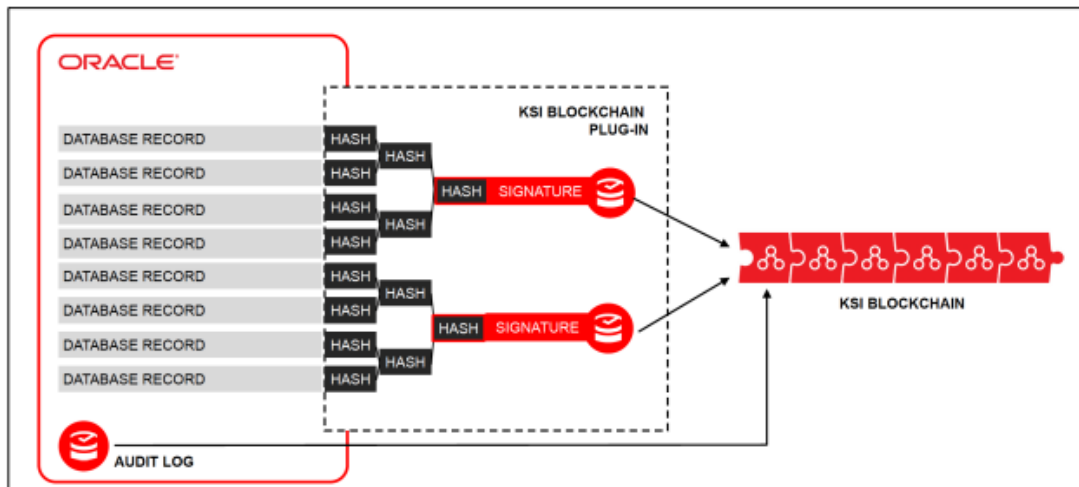


Figura 8: Esempio di KSI blockchain

KSI blockchain è sviluppata all'interno di singoli database statali per aggiungere sicurezza. Possiamo immaginare KSI blockchain come un sistema che registra un'istantanea del database locale e produce una chiave univoca.

KSI è formata da un'unica catena di *calendar hash* (CHC), ovvero un database distribuito attraverso l'infrastruttura. I record non possono essere cancellati ma solamente aggiunti. Ad esempio, quando un utente desidera inserire un dato all'interno del database, manda una richiesta al gateway. L'aggregator cluster registra l'hash del dato all'interno del calendar hash e ritorna all'utente la firma. Il token di firma fornisce informazioni come l'orario di firma, l'identità della firma e l'integrità del dato. KSI collega tutti gli hash dei file creati o modificati in uno slot temporale utilizzando un Merkle tree. La radice in cima al Merkle tree viene combinata con la radice del precedente slot temporale e viene scritto nel calendar hash.

Sempre nella pubblicazione di Guardtime [39] si afferma che in KSI blockchain, diversamente da Bitcoin, non è presente alcun algoritmo di consenso e i dati non vengono salvati, ma viene salvato il loro hash. Queste scelte implementative permettono a KSI un basso consumo di spazio e banda oltre ad una grande velocità. Allo stesso tempo riducono la trasparenza, dato che i dati vengono memorizzati in database locali.

Per quanto definito nel paragrafo 1.2, KSI non può essere considerata una vera blockchain poiché non è presente un meccanismo di consenso distribuito e non esistono comunicazioni P2P tra i nodi.

4.7 Voatz

Voatz [26] è una piattaforma nata nel 2015 per effettuare votazioni elettroniche in cui è possibile votare tramite un dispositivo mobile. La sicurezza del sistema è costituita dalle ultime tecnologie per smartphone e dell'immutabilità della blockchain. Da giugno 2016, Voatz ha partecipato a più di 50 elezioni.

Solo i più recenti smartphone come Apple, Samsung e Google sono supportati da Voatz. I dispositivi devono possedere alcune feature di sicurezza, come il riconoscimento dell'impronta digitale e il riconoscimento facciale, oltre ad altre proprietà di sicurezza basate sull'hardware. I sistemi operativi in modalità jailbroken o rooted non possono utilizzare Voatz poiché possiedono speciali privilegi che potrebbero compromettere la sicurezza del dispositivo.

L'autenticazione è composta da tre step:

1. scansione della patente o del passaporto;
2. video selfie del volto;
3. riconoscimento dell'impronta o del volto.

Quando l'elettore viene autenticato, l'applicazione verifica la corrispondenza tra selfie e la foto sul documento scansionato e permette all'utente di votare se esso è registrato nel database dei votanti.

Tipicamente all'apertura della votazione, l'utente riceve un bussolotto che viene spedito al server solo dopo che l'elettore ha espresso il voto e confermato la propria scelta. Tutte le informazioni vengono anonimizzate tramite una *mixnet* e registrate sulla blockchain. Una volta che il voto viene confermato sulla blockchain, l'utente riceve una notifica.

Voatz utilizza una permissioned blockchain, costruita sul framework HyperLedger blockchain di IBM [40]. Il bussolotto che ciascun elettore riceve contiene dei token, che corrispondono ai voti esprimibili. Il numero di token che un determinato elettore riceve è pari al numero di schede elettorali che esso avrebbe ricevuto in una votazione basata su carta. I bussolotti, una volta spediti, vengono verificati da molteplici server distribuiti chiamati "verifiers" o validating node. Il numero minimo di nodi validatori è 4, e può essere aumentato a 16 o 32.

Per quanto riguarda la proprietà di anonimità, dopo l'autenticazione, l'app Voatz cifra l'identità dell'elettore, collegando il telefono al votante tramite l'impronta digitale e rimuovendo ogni altra informazione di identificazione. In questo modo l'elettore può votare solo tramite quel dispositivo [26].

Alcuni ricercatori del MIT, tramite un processo di reverse engineering hanno ricostruito la piattaforma di Voatz e ne hanno studiato le proprietà di sicurezza. I ricercatori hanno concluso che chiunque sia in grado di compromettere il telefono di un votante può vedere, cancellare e modificare i voti. Inoltre chiunque controlli il server può osservare, alterare e aggiungere voti a suo piacimento [41].

4.8 VoteCoin

VoteCoin [27] è un'implementazione del protocollo di Zerocash, o meglio è una variante di Zcash in cui:

- non esistono le founder reward;
- non c'è slow start period, ovvero i miner ricevono l'intera ricompensa fin da subito;
- il wallet è multiplatforma.

Zcash ha due tipi di indirizzi:

- *transparent*, dove tutte le informazioni riguardo alle transazioni sono pubblicamente visibili sulla blockchain;
- *shielded*, dove non è possibile avere informazioni riguardo alle transazioni.

Ogni elettore ha un wallet e invia il proprio token al wallet del candidato. Vince il candidato che colleziona più coin. Quando si utilizza voteCoin per una votazione, trasferire un coin non significa solo votare ma anche trasferire valore, motivo per cui oltre ad un'elezione si può utilizzare voteCoin per campagne di crowdfunding. Questo rende costoso ogni tentativo di frode. Al termine della votazione, ogni candidato deve trasferire le monete ricevute ad un transparent address. In questo modo si rende pubblico il risultato delle elezioni e i fondi tornano a disposizione dell'organizzazione che ha istituito la votazione.

VoteCoin mira a far evolvere gli attuali sistemi elettorali con diversi scenari, come ad esempio dare la possibilità ad un cittadino di dividere il proprio voto tra più candidati. Oppure dare la possibilità di trasferire il proprio diritto di voto a qualcun'altro.

5 Sistema di voto basato su Monero

Come si è detto, attualmente non esiste un sistema di voto elettronico riconosciuto universalmente come valido. Questo poiché un sistema di voto deve soddisfare precisi requisiti di privacy e sicurezza. L'utilizzo della blockchain nella progettazione di un sistema di voto risulta particolarmente adatta poiché garantisce accessibilità, immutabilità e resilienza.

Alcuni paper propongono l'utilizzo di Ethereum [42, 43], in quanto questa blockchain permette la creazione di applicazioni decentralizzate. Sebbene sia possibile scrivere smart contract con Solidity, un linguaggio Turing-completo, Ethereum, come Bitcoin, non permette di garantire privacy e anonimato. Per questo motivo si è optato per utilizzare una blockchain attenta all'anonimato. Zcash e Monero sono due blockchain note per la loro propensione all'anonimato, anche grazie all'utilizzo di zero-knowledge proof. Le zero-knowledge proof utilizzate dalle due diverse blockchain sono descritte nel capitolo 3. Si noti come le ZK-SNARKs presenti in Zcash necessitano di una trusted ceremony e per questo motivo non risultano adatte ad un sistema di voto.

Dopo un'attenta analisi tra i due protocolli, è emerso che attualmente Monero risulta essere più adatto ad essere utilizzato in un sistema di voto. In primo luogo il protocollo di Monero è ben definito, e sebbene nel corso del tempo abbia subito delle evoluzioni, risulta essere stabile. Al contrario Zcash è più recente e deve essere ancora sviluppato per raggiungere la sua versione definitiva.

In secondo luogo il protocollo di Zcash richiederebbe molte più modifiche rispetto a quello di Monero per essere utilizzato in un sistema di voto. Infatti Zcash vuole creare una rete in cui, oltre alle normali transazioni Bitcoin, sia possibile effettuare delle transazioni nascoste. Inoltre la zero-knowledge proof presente in Zcash nasconde mittente, destinatario ed ammontare. Invece Monero, come descritto nel paragrafo 5.1, utilizza diversi meccanismi per nascondere il mittente, il destinatario e l'ammontare della transazione. Questa caratteristica permette una maggiore malleabilità del protocollo, utile per apportare le modifiche necessarie per utilizzare la blockchain nel sistema di voto. Infine sono stati condotti maggiori studi sulle vulnerabilità di privacy ed anonimato di Monero rispetto a Zcash. Questo poiché la percentuale di transazioni nascoste in Zcash è molto bassa.

Nella prima parte del capitolo, vengono spiegate le caratteristiche salienti della blockchain di Monero, illustrando in particolar modo i meccanismi che rendono questa blockchain attenta all'anonimato. Successivamente si descrivono le proprietà di sicurezza che un sistema di voto deve avere. Infine vengono analizzate le scelte implementative fatte per la modifica del protocollo di Monero e per la messa in opera dell'esperimento.

5.1 Monero

Monero [44] nasce nel 2014, periodo contraddistinto dalla nascita di diverse criptovalute. Con esse condivide l'utilizzo della blockchain e l'applicazione di schemi crittografici avanzati. La maggior parte delle blockchain pubbliche, come Bitcoin, memorizzano i dati in chiaro, per favorire la verificabilità delle transazioni da parte di tutti gli utenti. Al contrario Monero mira ad ottenere la privacy delle transazioni, per quanto riguarda il mittente, il destinatario ed il contenuto. Questo è possibile grazie a:

- ring signature;
- stealth address;
- bulletproof.

Monero viene sviluppato da un team ristretto con l'aggiunta di alcune persone; nel corso della sua storia, più di 500 sviluppatori hanno partecipato al progetto. Inoltre esiste una fervente community interessata al progetto e il Monero Research Lab [45], che effettua ricerche riguardanti il protocollo e l'analisi di eventuali vulnerabilità. Monero è un progetto in continua evoluzione, infatti circa ogni sei mesi vengono programmati degli aggiornamenti software che implementano nuove caratteristiche nella blockchain. Questi sono dei veri e propri hard fork e per questo motivo gli utenti devono assicurarsi di eseguire la versione corrente per continuare ed essere parte della rete di Monero.

Un'altra peculiarità di Monero è l'algoritmo per effettuare il mining. Infatti il 30 novembre 2019, o più precisamente dal blocco numero 1978433, RandomX è diventato l'algoritmo per il mining. Secondo gli sviluppatori cambiare periodicamente l'algoritmo permette di ottenere una migliore decentralizzazione della rete, evitando che venga sviluppato un hardware specializzato in una particolare computazione.

Il progetto è open-source e il codice è disponibile alla repository gitHub <https://github.com/monero-project/monero>. Per comprendere meglio il funzionamento di Monero e i suoi limiti, esiste un libro, Mastering Monero [46] e una playlist di video su You Tube, intitolata Breaking Monero [47] e prodotta da alcuni membri del team di sviluppo.

Il nome Monero è la traduzione in esperanto di moneta.

5.1.1 Crittografia

La crittografia di Monero si basa sull'utilizzo delle curve ellittiche, in particolare della curva ed25519 Twisted Edward [48]. Essa può essere espressa come:

$$-x^2 + y^2 = 1 - (121665/121666)x^2y^2$$

La curva è stata sviluppata da un team indipendente al NIST, e attualmente viene considerata dagli esperti come sicura.

Come già spiegato nel paragrafo 2.1, il problema del logaritmo discreto su curva ellittica consiste nel fatto che moltiplicando un punto per uno scalare molto grande, è computazionalmente difficile ottenere il valore dello scalare a partire dal risultato della moltiplicazione. Quindi l'operazione di moltiplicazione scalare viene considerata una funzione one-way.

Monero utilizza la moltiplicazione su curva ellittica per generare le chiavi pubbliche a partire da quelle private.

In Monero esistono quattro diversi tipi di chiavi:

- **private view key** (k^v): utilizzata per verificare il saldo di un wallet;
- **public view key** (K^v): utilizzata per creare indirizzi validi, stealth address;
- **private spend key** (k^s): utilizzata per firmare le transazioni, cioè per spendere monero;
- **public spend key** (K^s): utilizzata per verificare la firma delle transazioni.

La funzione di hash utilizzata in Monero è la funzione Keccak [49], progettata da un gruppo indipendente che ha vinto la competizione per definire lo standard SHA-3 [50]. La funzione di hash Keccak-256 produce un output di 32 byte.

Un buon *pseudo random number generator*, PRNG, deve introdurre molta entropia, per rendere l'output non predicibile. Quando si genera un wallet, il sistema operativo genera il seed. Monero applica poi ripetutamente la funzione di Keccak per produrre un output non predicibile (ciascun round di hash produce un output che viene utilizzato come input dal round successivo).

Esistono due approcci per generare le chiavi segrete:

1. generare in maniera casuale entrambe le chiavi private;
2. generare in maniera casuale un seed.

Nel caso in cui venga generato il seed, esso può essere convertito in venticinque parole, di cui l'ultima viene utilizzata come checksum. Per convertire la stringa binaria in una frase di venticinque parole si utilizza una conversione in base 1626, ovvero per ogni lingua disponibile esiste un dizionario formato da 1626 parole grazie al quale è possibile effettuare la conversione. La private view key viene generata dall'hash del seed e tramite l'applicazione della funzione `sc.reduce32` si ottiene un valore compatibile con la curva ellittica. Mentre la private spend key non è altro che il seed.

Indice	Grandezza in byte	Descrizione
0	1	Identifica il tipo di rete e il tipo di indirizzo
1	32	public spend key
33	32	public view key
64	4	checksum creata dai primi 4 byte del risultato della funzione di Keccak applicata ai precedenti 64 byte

Tabella 2: Composizione di un indirizzo nella blockchain di Monero

Per generare le rispettive chiavi pubbliche, le due chiavi private vengono moltiplicate per un generatore, ovvero un punto G fissato, appartenente alla curva ed25519. Si noti come le chiavi private non sono punti della curva, mentre le chiavi pubbliche sì. Normalmente la public key viene rappresentata da una stringa formata dalla combinazione tra la coordinata y del punto e la coordinata x . In particolare si considera il valore di y ordinato in little endian (il primo byte della stringa è posizionato a destra) e si sostituisce il bit più significativo di y con il bit meno significativo del valore di x . Dato che il processo per la generazione delle chiavi a partire da un seed è deterministico, memorizzando il seed è possibile cambiare anche il tipo di wallet.

La tabella 2 mostra come è strutturato l'indirizzo di un wallet. A causa della lunghezza della stringa, spesso l'indirizzo viene convertito in base 58, oppure per facilitarne lo scambio, viene rappresentato da un qrCode.

Qualora due utenti ricevano da un'entità terza lo stesso indirizzo, se colludono possono scoprire che il destinatario della transazione è lo stesso. Per questo motivo solitamente vengono utilizzati i *subaddress*. Un subaddress è un indirizzo appartenente al wallet che non può essere collegato a nessun altro subaddress o all'indirizzo principale. Un wallet può generare un grandissimo numero di subaddress utilizzando la stessa public view key, la public spend key e un indice differente i . Per ogni subaddress esiste una coppia di chiavi pubbliche (K_i^s, K_i^v) e private (k_i^s, k_i^v) , così generate

$$K_i^s = H(k_0^v, i)G + K_0^s$$

$$K_i^v = k_0^v K_i^s$$

dove H è la funzione di Keccak, k_i^s e k_i^v sono le chiavi private del subaddress i (rispettivamente spend key e view key), mentre le chiavi con indice 0 corrispondono alle chiavi del wallet. Sebbene l'indirizzo di un subaddress sia costruito esattamente come un normale indirizzo, nella mainnet i subaddress hanno un prefisso di 0X42, cioè la cifra 8.

5.1.2 Transazione

Un utente possiede due coppie di chiavi pubbliche e private (k^v, K^v) e (k^s, K^s) . Si consideri k^v come la private view key e k^s la private spend key. Il wallet utilizza la private view key per determinare se è il destinatario degli output di una transazione, mentre la private spend key permette di firmare gli output di cui è destinatario e quindi di utilizzarli come input in altre transazioni.

Diversamente da Bitcoin, l'indirizzo di un wallet non viene mai utilizzato come destinatario di una transazione, altrimenti un osservatore esterno che conosce gli indirizzi pubblici potrebbe collegare gli output ai rispettivi destinatari. Per questo motivo per ogni output bisogna generare uno *stealth address*, chiamato anche *one-time address*. Per creare uno stealth address si utilizza il protocollo di Diffie-Hellman [51]. Sia $H_n(x) = sc_reduce32(Keccak(x))$

Nel caso di una transazione semplice (un input e un output) tra Alice e Bob:

1. Alice genera un numero casuale $r \in \mathbb{Z}_l$ e calcola la one-time public key

$$K^o = H_n(rK_B^v)G + K_B^s$$

2. Alice imposta K^o come indirizzo di pagamento, aggiunge il valore rG ai dati della transazione e la invia alla rete;
3. Bob riceve come dati K^o e rG . Esso può calcolare $k_B^v rG = rK_B^v$ e anche

$$K_B'^s = K^o - H_n(rK_B^v)G$$

Quando si accorge che $K_B'^s = K_B^s$ esso sa che è indirizzata a lui;

4. le chiavi one-time per l'output sono:

$$\begin{aligned} K^o &= H_n(rK_B^v)G + K_B^s = (H_n(rK_B^v) + k_B^s)G \\ k^o &= H_n(rK_B^v) + k_B^s \end{aligned}$$

Alice non può computare k^o poiché dovrebbe conoscere la private spend key di Bob k_B^s oppure risolvere il logaritmo discreto $K_B^s = k_B^s G$.

In una transazione multi-output, il mittente genera comunque un solo numero casuale r . Per assicurarsi che tutti gli indirizzi di output siano diversi, anche se generati dal medesimo indirizzo, Monero utilizza un indice t . Esso viene concatenato a rG (conosciuta anche come *transaction public key*) prima che sia utilizzato come input della funzione di hash. Questo permette la creazione di stealth address unici. Quindi:

$$\begin{aligned} K_t^0 &= H_n(rK_t^v, t)G + K_t^s = (H_n(rK_t^v, t) + k_t^s)G \\ k_t^0 &= H_n(rK_t^v, t) + k_t^s \end{aligned}$$

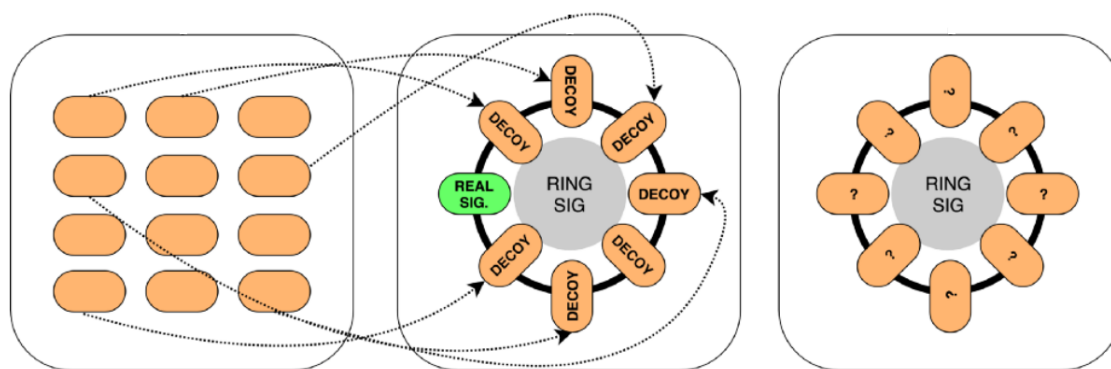


Figura 9: Meccanismo della ring signature

Per differenziare gli output, un destinatario può chiedere al mittente di inserire un payment Id nei metadati. Questa pratica è comunque sconsigliata poichè rompe l'uniformità delle transazioni. Al posto dei payment Id è possibile utilizzare differenti subaddress.

In una transazione Monero il valore dei diversi output viene nascosto in due diversi modi. Nel campo *ecdhInfo* l'importo viene cifrato tramite l'utilizzo del cosiddetto *shared secret*, ovvero un punto della curva ellittica calcolabile solo dal mittente e dal destinatario. Questo punto viene calcolato dalla moltiplicazione scalare tra il numero casuale r , chiamato anche *transaction private key* e la public view key del destinatario. In questo modo solo il destinatario della transazione può decifrare questo campo.

Affinché un osservatore esterno possa verificare la correttezza della transazione, cioè controllare che la somma degli output sia uguale alla somma degli input e delle fee e che i valori degli input e degli output siano positivi o nulli, si utilizza la bulletproof (paragrafo 3.3).

5.1.3 Ring Signature

Per celare il mittente di una transazione, Monero utilizza la ring signature. In particolare, questo meccanismo permette di nascondere quali output vengono spesi in una transazione (figura 9). Ad esempio Maria vuole inviare una transazione Monero. Il suo wallet prima sceglie l'output da spendere e poi, dopo aver selezionato dalla blockchain altri output, aggiunge le loro chiavi pubbliche alla ring signature come *decoy*. Ogni partecipante della rete è in grado di verificare che un output è stato speso, ma i decoy e l'output speso sono crittograficamente indistinguibili. Per evitare attacchi di doppia spesa, per ogni output speso si calcola una key image che lo identifica univocamente mantenendo comunque l'anonimato.

Le ring signature sono un'evoluzione delle group signature, in cui un group manager

sceglie i membri del gruppo ed in caso di disputa può rivelare il mittente della transazione. Ciascuna firma viene generata grazie all'utilizzo della chiave pubblica e privata di un output e da un insieme di chiavi pubbliche scelte in maniera pseudo-casuale. Le chiavi pubbliche di una ring signature scelte in maniera casuale sono anche dette *mixin*. Una ring signature deve rispettare tre caratteristiche:

- *signer ambiguity*: un osservatore deve essere in grado di determinare se è un membro dell'anello (ad eccezione di una piccolissima probabilità) ma non quale membro. Monero utilizza questa caratteristica per offuscare chi ha creato la transazione;
- *linkability*: se una chiave privata viene utilizzata per firmare due messaggi differenti allora i messaggi risultano collegati. Monero utilizza questa caratteristica per evitare attacchi di doppia spesa;
- *unforgeability*: nessun attaccante può creare una firma valida senza essere in possesso della chiave privata.

Sia m il messaggio da firmare e $R = \{K_1, K_2, \dots, K_n\}$ l'insieme delle chiavi pubbliche, k la chiave privata corrispondente alla chiave pubblica $K_\pi \in R$, dove π è un indice segreto (si ricordi che $K_\pi = k_\pi G$). Si assume l'esistenza di due funzioni di hash, H_n e H_p che ritornano rispettivamente un intero da 1 a l e un punto della curva ellittica. In Monero, $H_n(x) = sc_reduce32(Keccak(x))$ [52]. Si eseguono poi le seguenti operazioni:

1. si calcola una key-image $\tilde{K} = k_\pi H_p(K_\pi)$;
2. si genera un valore casuale $\alpha \in_R Z_l$ e dei numeri casuali per ogni “firma falsa” $r_i \in_R Z_l$ per $i \in \{1, 2, \dots, n\}$ ma escludendo $i = \pi$;
3. si calcola $c_{\pi+1} = H_n(m, [\alpha G], [H_p(K_\pi)])$;
4. per ogni $i = \pi + 1, \pi + 2, \dots, n, 1, 2, \dots, \pi - 1$ si calcola, ricordando che la posizione $n + 1$ corrisponde a 1

$$c_{i+1} = H_n(m, [r_i G + c_i K_i], [r_i H_p(R) + c_i K])$$

5. si definisce la risposta reale r_π tale che $\alpha = r_\pi + c_\pi k_\pi \pmod{l}$

Per ogni $i = 1, 2, \dots, n$ la firma i -esima utilizza il numero casuale e la firma definita nella posizione $i - 1$. Per nascondere la posizione della chiave scelta, si calcola $r_\pi = \alpha - c_\pi k_\pi \pmod{l}$ in modo che anche $c_{\pi+1}$, la prima firma generata, per un osservatore esterno risulti essere uguale alle altre. La ring signature è composta dalla firma $\sigma(m) = (c_1, r_1, r_2, \dots, r_n)$, dalla key-image \tilde{K} e dall'insieme di chiavi pubbliche R .

Per verificare una firma bisogna ricalcolare c_1 . Effettuare la verifica significa provare che $\sigma(m)$ è una firma valida creata da una chiave privata corrispondente ad una delle chiavi contenute in R . I passaggi sono:

1. controllare che $l\tilde{K} = 0$;
2. per $i = 1, 2, \dots, n$ computare iterativamente ricordando che la posizione $n + 1$ corrisponde a 1

$$c'_{i+1} = H_n(R, \tilde{K}, m, [r_i G + c_i K_i], [r_i H_p(R) + c_i \tilde{K}])$$

3. se $c'_1 = c_1$ allora la firma è valida. Si noti che c'_1 è l'ultimo termine calcolato.

Dalla nascita di Monero e fino a circa il 2016, le ring signature erano formate con un numero a piacere di mixin, anche 0. Nel caso in cui si decideva di utilizzare zero mixin, l'indirizzo del mittente era chiaro a tutti (0-decoy attack). Questo fatto, oltre ad essere contrario alle caratteristiche di anonimato e privacy insite in Monero, comportava una diminuzione della unlinkability nelle altre transazioni. Infatti, se in una ring signature sono compresi solo indirizzi 0-decoy, l'output speso risulta essere facilmente calcolabile. Per questo motivo, durante gli hard fork successivi è stato imposto un numero minimo di indirizzi da includere nella ring signature. Attualmente il numero minimo è di 11 mixin.

Un altro fatto importante che impatta sull'efficacia della ring signature è la scelta dei mixin. Inizialmente l'algoritmo di scelta delle firme selezionava solo indirizzi generati precedentemente alla transazione in essere. Questo però era molto limitante, poiché per trovare l'indirizzo speso era sufficiente determinare quale indirizzo fosse stato prodotto per ultimo. Come il resto del sistema, anche l'algoritmo di scelta delle firme è in continua evoluzione e miglioramento. Attualmente l'algoritmo seleziona il maggior numero di mixin in quella che viene definita *recent zone* (cioè indirizzi generati in un lasso di tempo precedente o successivo alla creazione della transazione interessata).

Logicamente sussiste una certo legame tra numero di indirizzi richiesti in una ring signature e algoritmo di scelta dei mixin. Maggiori sono il numero di firme e minore è l'importanza dell'algoritmo di scelta dei mixin. Sebbene si continui a migliorare l'algoritmo di selezione degli indirizzi, non è pensabile che esso possa prevenire tutte le possibili euristiche utilizzabili per individuare gli output spesi.

Infine bisogna considerare la grandezza della ring signature. La ring signature può essere fatta con centinaia o migliaia di decoy. In generale, una ring signature con più decoy è più sicura di una con meno decoy, ma bisogna fare alcune precisazioni. Infatti, bisogna fare delle considerazioni quando si decide di utilizzare un maggior numero di decoy:

- grandezza della firma maggiore, con la necessità di spendere più fee affinché sia scelta dai miner;
- numero maggiore di informazioni;

- eventuali scelte come ad esempio 100 o più firme, potrebbero diversificare la ring signature rispetto al comportamento generale.

Quindi non è detto che ad un numero maggiore di firme corrisponda una ring signature più sicura. Per questo motivo si consiglia di utilizzare il numero minimo di firme previste dal protocollo, in modo da uniformare il più possibile le transazioni.

5.2 Sistema di voto

Esistono differenti sistemi di voto. Alcuni vengono utilizzati nei reality show o nei contest, altri vengono adottati nelle associazioni e nei consigli di amministrazione, altri ancora vengono impiegati per le elezioni politiche. Da un punto di vista tecnico, possiamo suddividere i sistemi di voto in due categorie: quelli in loco, in cui l'elettore si reca fisicamente in un luogo ed esprime il proprio voto, e i sistemi da remoto, dove è possibile partecipare alla votazione indipendentemente dal luogo in cui ci si ritrova.

In un mondo fortemente improntato alla globalizzazione, considerando la necessità di evitare assembramenti a causa della recente emergenza pandemica, la creazione di un sistema di voto da remoto assume una maggiore importanza. Per questi motivi si è scelto di creare un sistema di voto da remoto.

Indipendentemente dal tipo di sistema, ogni votazione ha le proprie regole e i propri requisiti. Dato che l'impiego della blockchain richiede uno sforzo a livello di complessità, si è deciso di considerare la tipologia di votazione più importante, cioè quella inerente alle elezioni politiche.

Ora vengono definite le caratteristiche che il sistema di voto proposto deve rispettare. Si noti che il sistema di voto deve assicurare l'anonimato degli elettori. Inoltre non deve essere possibile conoscere l'andamento delle elezioni in tempo reale poiché potrebbe influenzare gli elettori che non hanno ancora espresso il proprio voto. Nella tabella 3 vengono descritti i requisiti tramite i quali viene valutato il sistema di voto [42].

I requisiti sono elencati in ordine decrescente di importanza. Si osservi che i requisiti con un impatto maggiore sono inerenti alla sicurezza e all'anonimato del sistema, mentre i requisiti con un impatto minore riguardano l'usabilità e la trasparenza dell'infrastruttura di voto. Inoltre si noti come i requisiti di mobilità e capacità di controllo e ripristino siano specifici per un sistema di voto elettronico.

Il sistema di voto utilizza la blockchain di Monero, con alcune modifiche necessarie per soddisfare i requisiti espressi. Per differenziare le due blockchain, il nome V-Monero indica la versione modificata di Monero e impiegata nel sistema di voto.

L'espressione di una scelta di voto in un sistema tradizionale viene sostituita da una transazione sulla blockchain. Per convenzione, 1 token di voto corrisponde a 1 monero.

Requisiti	Descrizione
Autenticità	Solo gli utenti con il diritto di voto possono esprimere il voto
Singularità	Ciascun votante può votare solo una volta
Anonimato	Deve essere impossibile associare il voto al votante
Integrità/immutabilità	I voti non possono essere modificati o distrutti
Non coercibilità	Nessun elettore può provare la scelta espressa nella votazione
Verificabilità	Chiunque, in maniera indipendente, deve essere in grado di verificare che tutti i voti siano stati correttamente conteggiati
Auditabilità / certificabilità	Il sistema di voto deve essere testato, verificato e certificato da operatori esterni
Mobilità	Il sistema di voto non deve limitare il luogo da cui viene effettuato il voto
Trasparenza	Il sistema di voto deve essere chiaro e deve trasmettere accuratezza, precisione e sicurezza all'elettore
Disponibilità	Il sistema deve essere sempre disponibile durante il periodo di voto
Accessibilità	Il sistema di voto deve essere accessibile alle persone con bisogni speciali e non deve richiedere specifiche abilità o attrezzature
Capacità di controllo /ripristino	Il sistema di voto deve identificare errori, problemi ed attacchi e recuperare le informazioni a partire dal point of failure

Tabella 3: Proprietà di un sistema di voto

Si suppone che le proprie chiavi private generate dal wallet non siano cedibili. Altrimenti chiunque potrebbe impersonificare qualsiasi persona. Questa affermazione viene discussa nel paragrafo 6.2, dove si parla del problema dell'identità digitale. In aggiunta, ciascuna sessione di voto è indipendente dalle altre e non condivide alcun dato. Quindi anche nel caso in cui l'organizzatore della votazione e i partecipanti siano gli stessi, ogni votazione ha una propria blockchain.

Nel sistema sono presenti diversi attori:

- *elettori*: coloro che hanno il diritto di partecipare all'elezione. Ogni elettore deve installare sul proprio smartphone un'applicazione che si occupa della generazione delle chiavi e della gestione del wallet. Sempre tramite l'applicazione è possibile verificare che il voto sia stato incluso nella blockchain;
- *amministratore*: la persona che si occupa di gestire le elezioni, come iniziare e terminare la fase di voto, calcolare i risultati. Solitamente è un rappresentante dell'organizzazione che indice le elezioni;
- *candidati*: corrispondono alle scelte esprimibili nella votazione. Ogni candidato deve avere un proprio view only wallet con un indirizzo pubblico. Un view only wallet è una possibilità esprimibile durante la generazione di un wallet ed indica che la private spend key non viene generata. In questo modo il wallet può solo ricevere fondi senza mai trasferirli;
- *custodi*: coloro che possiedono la private view key dei candidati. Essi si occupano di non divulgare le chiavi fino al completamento dell'elezione. Questo compito può essere svolto dall'amministratore stesso.

Per quanto riguarda la blockchain, la rete è costituita da diversi nodi certificati e gestiti dall'ente organizzatore. È possibile includere nella rete altri nodi, a condizione che questi siano gestiti da entità certificate. Infatti un remote node è in grado di ottenere molti metadati dalle transazioni provenienti dai software wallet. Quindi per motivi di privacy e sicurezza non è possibile aggiungere remote node a piacere. Inoltre, l'organizzazione del voto mette a disposizione uno o più explorer, utilizzabili da chiunque, per verificare che il voto venga conteggiato.

Il processo di voto può essere suddiviso in tre differenti fasi: registrazione, votazione e spoglio. La fase di registrazione è precedente alla votazione e ha una durata fissata. Durante la fase di registrazione, un elettore che desidera utilizzare il sistema di voto da remoto deve comunicare il proprio indirizzo all'organizzazione dell'elezione. A questo punto si presenta il problema dell'identità digitale. Infatti l'elettore dovrebbe dimostrare la propria identità e di essere l'unico a possedere effettivamente la coppia di chiavi private corrispondenti all'indirizzo inserito.

Per quanto riguarda il primo punto, la soluzione più semplice è effettuare l'autenticazione ad un portale gestito dall'ente che organizza le elezioni. Oppure bisognerebbe integrare nell'applicazione un servizio di identità digitale, come ad esempio SPID.

Invece per il secondo punto la soluzione è più complessa. Bisognerebbe assicurarsi che non sia possibile trasferire le chiavi private e che l'utilizzatore sia lo stesso che ha disposto la generazione del wallet. Questi problemi sono inerenti al problema dell'identità digitale, che sebbene collegato, è al di fuori dello scopo di questa tesi. Per questo motivo si è definito che le chiavi private sono personali e non cedibili.

Questa separazione tra fase di registrazione e fase di votazione comporta delle scelte implementative differenti nel sistema di voto. Un sistema di voto organizzato in questo modo ha il vantaggio di poter essere affiancato a un qualsiasi sistema di voto tradizionale.

Successivamente alla fase di registrazione e prima dell'inizio della fase di votazione, l'amministratore dell'elezione imposta i parametri dell'elezione e comunica al sistema di voto gli indirizzi degli elettori, in modo che il wallet amministratore invii i token per effettuare la votazione.

Durante la fase di votazione, un elettore può esprimere il proprio voto. Per prima cosa deve connettersi ad un remote node certificato. Non è necessario esprimere l'indirizzo del candidato scelto poiché l'applicazione conosce gli indirizzi di tutti i candidati. A questo proposito sono possibili delle varianti. Nel caso in cui la transazione Monero contenga un solo output, si potrebbe utilizzare i qrCode oppure OpenAlias [53] piuttosto che l'inserimento manuale dell'indirizzo. OpenAlias è uno standard che permette di sostituire l'indirizzo dei wallet Monero con un alias più semplice da ricordare. In questo caso l'applicazione sarebbe più trasparente. Dopo aver effettuato il voto, un utente può utilizzare un explorer certificato per verificare tramite transaction Id che il voto sia stato effettivamente conteggiato.

Infine, per lo spoglio dei voti bisogna utilizzare le private view key dei wallet dei candidati. Il conteggio dei voti viene fatto in maniera automatica, come in qualsiasi wallet. L'amministratore deve assicurarsi che le chiavi non vengano divulgate fino al termine della votazione. Per questo motivo si potrebbe introdurre la figura dei custodi, una persona o un'organizzazione che si impegna a non divulgare le private view key dei candidati.

5.3 Implementazione

In questo paragrafo viene descritto il lavoro che è stato svolto per la creazione di V-Monero, ovvero l'adattamento di Monero al protocollo di voto.

Il progetto di Monero, sebbene molto grande, appare ben strutturato. Per motivi di

efficienza e sicurezza, è scritto nel linguaggio c++. Oltre al codice inerente alla blockchain, all'interno della repository è presente il codice di un software wallet e del server RPC. Il Monero daemon, chiamato monerod, è il cuore dell'applicazione, e si occupa della gestione della blockchain. Monerod è interamente disaccoppiato dal wallet e non ha accesso alle chiavi private. Infatti è possibile eseguire monerod su un altro computer o nel cloud. Il server RPC è una parte dell'applicazione monerod e permette agli sviluppatori di interagire con la blockchain tramite chiamate a procedura remota (RPC). È possibile invocare le RPC tramite chiamate HTTP. Le risposte contengono dati in formato JSON.

Tutte le RPC disponibili sono documentate all'indirizzo <https://web.getmonero.org/resources/developer-guides/daemon-rpc.html>. Inoltre esistono delle Wallet RPC che permettono di interagire con il monero-wallet-rpc, un software che sostituisce l'interfaccia presente in simplewallet e monero-wallet-cli. Queste sono descritte all'indirizzo <https://web.getmonero.org/resources/developer-guides/wallet-rpc.html>.

5.3.1 Avvio

Per compilare e testare la blockchain di Monero è stato utilizzato il sistema operativo Linux Ubuntu 18.04. Prima di compilare Monero è necessario installare delle specifiche librerie. Quindi si esegue

```
sudo apt get update
```

per aggiornare il sistema operativo. Per installare le librerie bisogna lanciare il comando

```
sudo apt install git build-essential cmake libboost-all-dev miniupnpc  
libunbound-dev graphviz doxygen libunwind8-dev pkg-config libssl-dev  
libcurl4-openssl-dev libgtest-dev libreadline-dev libzmq3-dev  
libsodium-dev libhidapi-dev libhidapi-libusb0
```

Ora bisogna recarsi nel percorso in cui si vuole clonare Monero e si lancia il comando

```
git clone https://github.com/monero-project/monero.git
```

Una volta entrati nella directory si compila Monero eseguendo:

```
cd monero  
USE_SINGLE_BUILDDIR=1 make -j1
```

dove il parametro -j specifica il numero di core utilizzati per l'operazione di compilazione. È sconsigliato utilizzare tutti i core della macchina a disposizione. Il parametro USE_SINGLE_BUILDDIR=1 non è indispensabile per la corretta compilazione di Monero, si è deciso di utilizzarlo per facilitare la creazione di un programma che interagisca con la blockchain.

Per lanciare un nodo è sufficiente lanciare il comando

```
./monero/build/release/bin/monerod
```

Per comprendere il funzionamento di Monero e del server RPC, si è scelto inizialmente di eseguire due nodi della blockchain in locale. Grazie all'utilizzo di script bash che effettuavano chiamate HTTP al RPC server collegato ad uno dei due nodi, si è verificata la correttezza delle modifiche apportate.

5.3.2 Modifiche apportate

La prima modifica effettuata riguarda la versione di Monero. Come è stato detto nella sezione 5.1, ogni sei mesi Monero riceve un aggiornamento e subisce un hard fork. Si è scelto di utilizzare la release 14.01 di Monero (corrispondente alla release 11 del fork) poiché questa versione richiede una ring signature di almeno 11 indirizzi e obbliga ad utilizzare la bulletproof al posto della vecchia versione della ring confidential transaction. Per fare questo, è stato modificato il file `hardforks.cpp` in modo che a partire dal secondo blocco di V-Monero valga l'undicesima versione del fork di Monero.

Si ricordi che Monero è una blockchain attenta all'anonimato e, come spiegato nella sezione 5.1, esistono diversi meccanismi per celare il mittente e il destinatario di una transazione. A differenza di Bitcoin o Monero, in V-Monero i partecipanti non sono tutti uguali. Infatti esistono tre differenti ruoli: l'amministratore, gli elettori e i candidati. Le transazioni eseguite dall'amministratore devono essere visibili a tutti, per una questione di trasparenza del sistema. In particolare sono noti la sua private view key e l'indirizzo in modo che chiunque abbia accesso alla blockchain possa riconoscere le transazioni eseguite. A questo punto si è affrontata la modifica principale, ovvero come limitare il trasferimento dei token in modo che gli elettori possano votare solo i candidati. In un certo senso questo problema ricorda il problema della doppia spesa poiché anche in questo caso bisogna limitare le transazioni.

La prima soluzione è quella di identificare il destinatario delle transazioni in modo da assicurarsi che un elettore possa trasferire il proprio token solo ad un candidato e non anche ad un altro elettore. Per fare ciò serve rendere nota a tutti le private view key dei candidati. Si è scelto di scartare questa soluzione poiché la conoscenza delle private view key dei candidati avrebbe comportato la possibilità di tracciare l'andamento delle votazioni in tempo reale.

A questo punto si è scelto di cambiare approccio al problema. Anziché distinguere tra wallet degli elettori e wallet dei candidati, si è deciso di limitare la trasferibilità del token. In V-Monero un token può appartenere a una delle categorie: spendibile o valido. Un token spendibile è un token inviato dall'amministratore. Un elettore che impiega un token spendibile nella generazione di una transazione produce un token valido, ovvero un token che è stato trasferito una sola volta. Questo token può essere inviato ad un qualsiasi partecipante della rete, potenzialmente anche ad un indirizzo appartenente ad un altro elettore. Come risulterà chiaro in seguito, avremo una limitazione nella trasferibilità del token. Un token valido non può essere speso e quindi diventa inutilizzabile

in un'altra transazione.

La ring signature di una transazione effettuata dall'amministratore è composta da soli indirizzi appartenenti all'amministratore. In questo modo è facilmente identificabile il mittente della transazione. Gli output generati da questo tipo di transazione sono riconosciuti dal sistema come spendibili. Un elettore quando vuole effettuare una transazione per votare, sceglie come decoy per la ring signature solo indirizzi spendibili. Questo poichè qualora ad un nodo venga sottoposta una ring signature composta diversamente, come ad esempio con un token valido, la transazione viene rifiutata. In questo modo si limita la trasferibilità del token. Ad esempio, se Alice invia il proprio token a Bob, l'output viene identificato come valido. Se Bob prova a spendere questo token, esso deve per forza essere incluso come input nella ring signature. Un nodo appartenente a V-Monero rifiuterà la transazione.

Nel dettaglio, ciascun output presente in una transazione viene identificato da un `global_index`, ovvero un indice globale che numera gli output per ordine di creazione. Se l'output è stato creato con una transazione anonima, il suo valore dichiarato è zero. In V-Monero, ad eccezione delle coinbase, tutte le transazioni sono anonime e quindi tutti gli output hanno valore zero.

Il codice che gestisce la logica della blockchain è posizionato nel file `blockchain.cpp`. In questo file, è stato creato un metodo `get_voting_transactions`. Questa funzione ha il compito di scansionare l'intera blockchain ispezionando tutte le transazioni appartenenti all'amministratore, per identificare gli indirizzi e gli indici globali degli output che possiede. Inoltre questo metodo individua anche gli indirizzi e indici globali degli output spendibili, ovvero prodotti da una transazione generata dal wallet dell'amministratore. Si osservi che se tra i diversi destinatari di una transazione inviata dall'amministratore c'è uno stealth address che gli appartiene, l'output associato non viene riconosciuto come spendibile poiché i token spendibili sono considerati tali solo se sono posseduti dagli elettori.

Inoltre è stata aggiunta la funzione `check_voting_validity` che verifica la corretta composizione della ring signature. Nel codice, in questa funzione è presente la logica che limita la trasferibilità del token, cioè evita che un output valido sia inserito nella ring signature. Per fare questo si va a ispezionare il campo `key_offsets`, ovvero un array contenente gli indici globali di tutti gli output coinvolti nella ring signature.

Il file `wallet2.cpp` gestisce il wallet. Da quanto si evince dal file, ora viene descritto il flusso logico del wallet. Una volta caricate le chiavi, il wallet ricerca, per ogni sub-address, gli output non spesi, in modo da calcolare il saldo disponibile. In seguito, a partire dall'output più vecchio, ne cerca uno che abbia un valore pari o superiore all'ammontare richiesto dalla transazione da creare, comprensivo di fee. A questo punto

il wallet deve selezionare i decoy per generare la ring signature. In Monero, La chiamata RPC `get_out_histogram` restituisce le informazioni sugli output di pari valore (si ricorda che se la transazione è fatta con ringCT o bulletproof, il valore è 0). Questa chiamata non solo identifica il numero di output, ma li suddivide in categorie, come ad esempio recenti o post fork. Questa suddivisione viene sfruttata dall'algoritmo per la scelta delle firme. Si ricorda che nella versione 14.01 di Monero, per comporre la ring signature, il wallet deve scegliere almeno altri 10 output. Malgrado ciò, il wallet richiede l'esistenza di un numero minimo di output disponibili molto maggiore (nei test effettuati, 67). Se non ci sono sufficienti output, il wallet termina l'operazione di creazione di una transazione con un errore. Conoscendo il numero totale di output con lo stesso ammontare, da questo gruppo bisogna selezionarne un sottoinsieme. Per fare ciò, viene utilizzata una distribuzione triangolare sugli indici globali degli output. In questo modo si ottengono gli indici globali da richiedere al demone di un full node. Per nascondere il proprio output, nella richiesta gli output vengono ordinati in base al loro indice globale. Logicamente tra questi indici è presente anche quello dell'output da spendere.

A questo punto si effettua la chiamata RPC `get_outs.bin` che ritorna per ogni output una serie di informazioni, come il blocco in cui è stato creato e la sua chiave pubblica. Prima di utilizzare queste informazioni, il wallet verifica che nella risposta sia presente l'output scelto, altrimenti il full node viene considerato inaffidabile e il wallet termina il processo di creazione. Tra gli output restituiti, bisogna sceglierne 10. Per la scelta, viene data priorità agli output già presenti in altre ring signature. Se non si dovesse raggiungere il numero di output necessari, i restanti vengono scelti in maniera casuale tra quelli restituiti.

In V-Monero non viene stravolto il flusso dell'esecuzione. Nel codice del wallet è stata aggiunta una RPC `get_voting_outputs` che accetta un parametro opzionale `role`. Se il campo `role` è `admin`, vengono ritornati gli indici globali delle transazioni appartenenti all'amministratore. Altrimenti vengono restituiti gli indici di tutti gli output spendibili. Anziché impiegare gli indici globali di tutte le transazioni salvate nella blockchain, si utilizzano gli indici della risposta RPC appena descritta. Nel caso in cui si tenti di spendere un output non spendibile, il wallet genera un errore e termina la creazione della transazione.

5.3.3 Ulteriori modifiche necessarie

Nel paragrafo precedente sono state spiegate le modifiche apportate alla versione di Monero utilizzata nell'esperimento. Esistono però altre modifiche che permettono di garantire un maggiore livello di privacy. Sebbene le transazioni siano nascoste grazie all'utilizzo della ring signature, dello stealth address e della bulletproof, esistono una grande quantità di metadati che possono essere impiegati per identificare una transazione. Quindi bisognerebbe evitare tutti quei casi che rompono l'uniformità delle

transazioni.

In una transazione Monero è possibile specificare un campo payment Id con l'obiettivo di identificare un ipotetico pagamento. L'utilizzo del payment Id rimane comunque una pratica sconsigliata in tutta la comunità di Monero, poiché svolge lo stesso compito di un subaddress con la differenza che risulta visibile e identificabile da chiunque. La versione di V-Monero deve rifiutare tutte quelle transazioni che prevedono l'utilizzo di un payment Id poiché potrebbero infrangere la segretezza del voto.

Un altro metadato utilizzabile per identificare una transazione potrebbe essere il numero di decoy utilizzati nella ring signature. Infatti Monero prevede un numero minimo di decoy per comporre la ring signature, ma non un numero massimo. Solitamente si utilizza il numero minimo richiesto. Poiché qualsiasi transazione con un numero di firme diverso potrebbe differenziarsi dalle altre, la versione di V-Monero deve controllare che il numero di mixin utilizzati sia pari a undici.

Infine, anche il numero destinatari potrebbe rompere l'uniformità delle transazioni. Per questo motivo le transazioni provenienti dall'amministratore avranno un numero di stealth address fissato, mentre nel caso delle transazioni prodotte dagli elettori, il numero di output deve essere pari al numero di candidati più uno, ovvero l'indirizzo utilizzato per il resto.

In un sistema di voto, a differenza di un sistema utilizzato per lo scambio di valore, i miner non hanno necessità di ottenere alcuna fee poiché il coin non ha alcun valore intrinseco e non può essere utilizzato. Si è deciso comunque di non rimuovere le fee per evitare un qualsiasi attacco DoS. Se la blockchain permettesse di inviare transazioni a costo zero, cioè senza fee, si potrebbero inviare transazioni senza muovere alcun coin, generando potenzialmente un numero illimitato di transazioni da validare e confermare.

Come è stato detto nella sezione 5.1 esistono differenti algoritmi di consenso; quello utilizzato da Monero si basa sulla proof of Work. Nel caso di un sistema di voto, un token non ha alcun valore intrinseco e quindi non è presente alcun incentivo per il miner che convalida un blocco. Inoltre, come è stato suggerito nella sezione 5.2 per motivi di privacy non è possibile aggiungere un nodo qualsiasi alla rete, quindi non tutti possono validare un blocco. Per questi motivi la proof of Work può essere sostituita dalla proof of Authority, una particolare proof of Stake. La proof of Authority prevede che nella rete esistano dei nodi certificati e solo loro possano validare i blocchi [54]. Per favorire una maggiore decentralizzazione della rete, solitamente si impone che un nodo authority non possa validare due blocchi consecutivi. Nel caso di un sistema di voto, è l'organizzatore della votazione che si occupa della gestione dei nodi authority. Questo tipo di algoritmo di consenso viene già utilizzato in Quorum e VeChain.

Al termine della sessione di voto, per certificare l'integrità e l'immutabilità della votazione, l'amministratore deve effettuare l'hash della blockchain e notarizzarla su una o più blockchain pubbliche. Per notarizzare un'informazione, spesso si utilizza il campo `OP_RETURN` presente nelle transazioni Bitcoin. Inoltre per assicurare un maggior grado di sicurezza e immutabilità, la pratica della notarizzazione potrebbe essere svolta periodicamente durante la sessione di voto [54].

5.3.4 Esperimento

Per eseguire una sperimentazione del sistema di voto, è necessario l'utilizzo di un software wallet da parte degli utenti o elettori. In più, all'interno di questa applicazione è necessario che sia presente un meccanismo di autenticazione che garantisca l'identità dell'utilizzatore.

Tra i diversi software wallet sviluppati per Monero, si è scelto di modificare Monerujo (<https://github.com/m2049r/xmrwallet>), un progetto open source. Uno svantaggio di questo software wallet è la disponibilità per i soli dispositivi Android. Prima di effettuare qualsiasi modifica, si è verificata la compatibilità dell'applicazione con la versione di Monero da utilizzare. Sebbene la repository su gitHub sia stata aggiornata per soddisfare i più recenti aggiornamenti di Monero, sono stati riscontrati degli errori nella fase di compilazione dell'applicazione. Infatti risulta un'incompatibilità tra le versioni delle librerie utilizzate. Purtroppo la documentazione risale al 2018 e non è stata aggiornata.

Per questi motivi si è scelto di produrre un backend che oltre a svolgere i diversi compiti dell'amministratore, gestisca anche i wallet degli elettori. Lo svantaggio di questa soluzione è quello di non poter verificare le proprietà di sicurezza dell'applicazione e del sistema operativo. Inoltre in questo modo viene a crearsi un single-point-of failure, in un sistema basato su blockchain, per sua stessa natura decentralizzata. Rimane comunque possibile verificare il funzionamento della blockchain e delle modifiche apportate al protocollo.

Al contrario il vantaggio di questa soluzione è un'esperienza d'uso più semplice da parte dell'utente, infatti, non è necessario installare alcuna applicazione ed è sufficiente fruire di una connessione ad Internet per effettuare la votazione. In questo caso, tramite le chiamate al server RPC, il backend utilizza il software wallet presente all'interno della repository di V-Monero, modificato come spiegato nel paragrafo 5.3.2.

L'implementazione del sistema è composta da un server Node.js che svolge i compiti dell'amministratore nella fase di preparazione della votazione e in quella di spoglio, gestisce il mining e i diversi wallet degli elettori. Per memorizzare le informazioni inerenti alla votazione e ai wallet, viene utilizzato MongoDB, un database NO-SQL. Per la

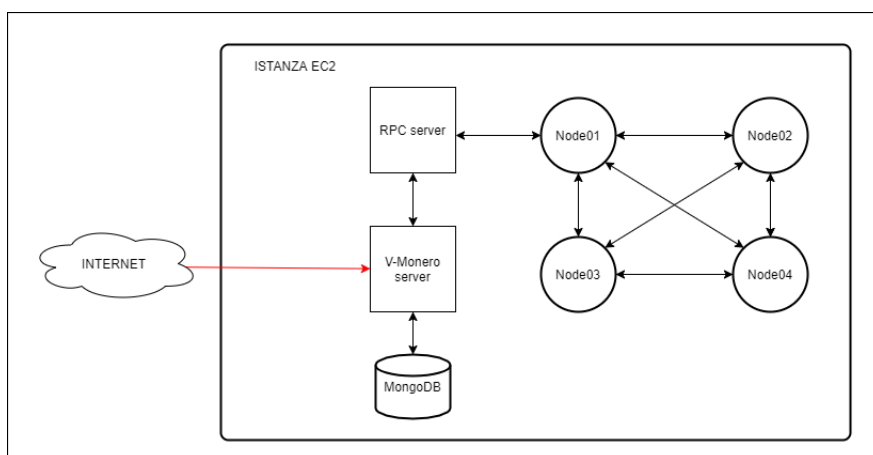


Figura 10: Architettura del sistema di voto

gestione della blockchain è previsto l'utilizzo di 4 nodi, e del server RPC. Il tutto viene eseguito in cloud, grazie all'utilizzo di AWS. La figura 10 mostra l'architettura del sistema di voto.

Amazon Elastic Compute Cloud (Amazon EC2) è un servizio Web che fornisce capacità di elaborazione sicura e scalabile nel cloud. Mediante l'interfaccia Web di Amazon EC2 è possibile ottenere e configurare la potenza di calcolo in modo semplice e immediato. L'utente ha il controllo completo delle proprie risorse informatiche, che possono essere eseguite nell'ambiente di elaborazione altamente efficiente di Amazon.

Un'Amazon Machine Image (AMI) è un modello che contiene una configurazione software (ad esempio un sistema operativo, un server di applicazioni e le applicazioni). Da un'AMI, è possibile avviare un'istanza, ovvero una copia della AMI in esecuzione come server virtuale nel cloud (figura 12).

Durante la configurazione della AMI è stato scelto Ubuntu 18.04 come sistema operativo. Come tipologia di istanze è stata scelta quella T3a poiché offre un buon compromesso tra prestazioni e costi. In più, è possibile personalizzare il numero di vCPU all'avvio dell'istanza. Questa caratteristica risulta fondamentale poiché compilare V-Monero richiede un maggior quantitativo di risorse rispetto alla semplice esecuzione del sistema. Infatti per compilare il codice in c++ è stato necessario utilizzare una T3a-medium, con 2 vCPU (ogni vCPU è un thread di un core AMD EPYC) e 4GB di RAM. Mentre per l'esecuzione del sistema è sufficiente una T3a-small, che ha le stesse caratteristiche ad eccezione di un quantitativo di RAM minore, pari a 2GB. Come Storage viene utilizzato

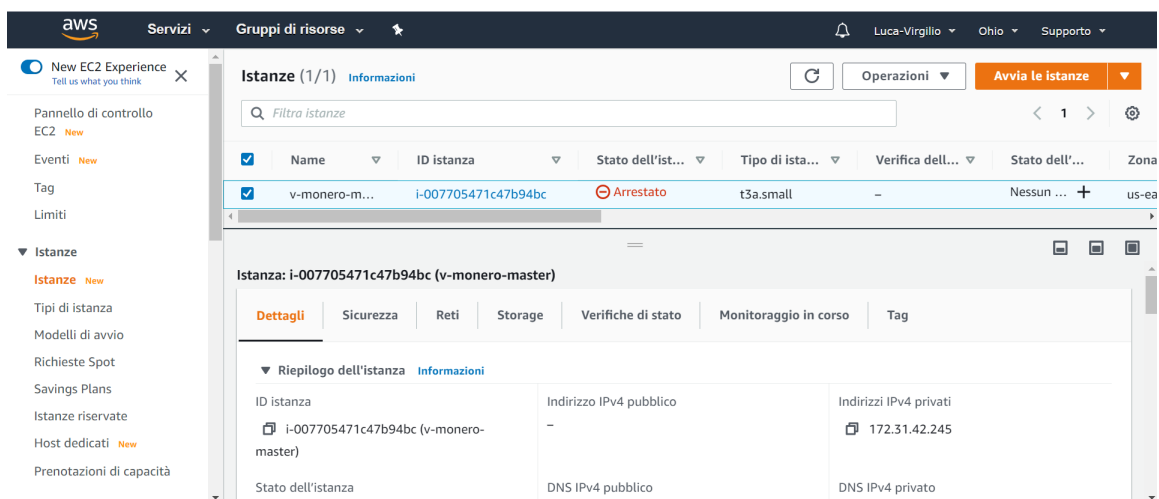


Figura 11: Dashboard del servizio AWS

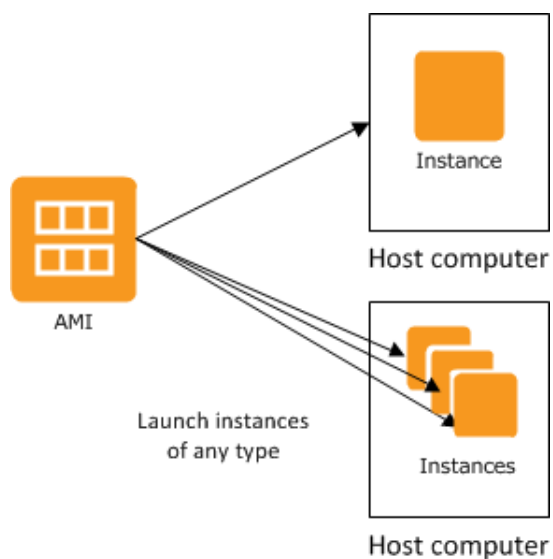


Figura 12: Istanze e AMI

EBS, ovvero uno spazio di archiviazione indipendente dal tipo di istanza e persistente fino alla cancellazione della AMI.

L'unica condizione presa in considerazione per la scelta della Amazon Region è la disponibilità di istanze T3a. Come si evince dalla figura 11, la regione scelta è stata Ohio.

Una o più istanze EC2 possono appartenere ad uno stesso security group. L'AWS Security Group è un tool messo a disposizione per una gestione flessibile delle istanze EC2. Esso è responsabile della sicurezza delle applicazioni e controlla il traffico in uscita ed in entrata, come un firewall. Per stabilire una connessione ssh con l'istanza EC2 è necessaria la creazione di una coppia di chiavi, come previsto nella crittografia asimmetrica. Quindi come prima regola d'ingresso, il security group prevede l'apertura della porta 22 (quella utilizzata da ssh) per il protocollo TCP. Non è stato specificato alcun indirizzo IP come sorgente. Come seconda regola d'ingresso è stata resa accessibile la porta 80 (quella utilizzata dal protocollo HTTP) da qualsiasi indirizzo IP, in modo che il frontend sia fruibile da chiunque.

Come prima cosa, sulla AMI è stato installato screen tramite il comando

```
sudo apt-get install screen
```

Screen è un multiplexer di terminali. Normalmente quando ci si connette ad una macchina in remoto, si ha a disposizione una sola finestra e qualora cada la connessione il lavoro svolto viene perso. Tutto questo non accade grazie a screen, poiché permette di creare delle sessioni in cui è possibile aprire molteplici terminali virtuali. Inoltre i processi in esecuzione in screen continuando ad essere eseguiti anche se la finestra non è visibile o la connessione ssh è terminata.

Per vedere il riassunto delle sessioni in corso si utilizza il comando

```
screen -list
```

Per creare una nuova sessione si usa il comando

```
screen -S nomeSessione
```

Quando si è all'interno di una sessione, è possibile uscire premendo la combinazione di tasti

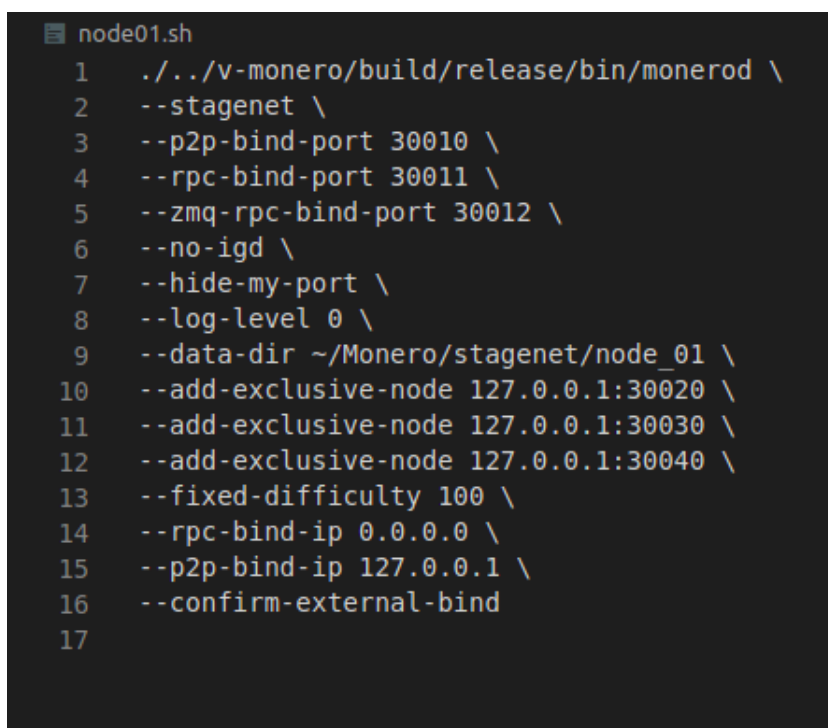
```
ctrl+a ctrl+d
```

per rientrare in una sessione, se questa è "attached" si usa il comando

```
screen -dr nomeSessione
```

altrimenti se la sessione è "detached" si utilizza

```
screen -r nomeSessione
```



```
node01.sh
1  ../../v-monero/build/release/bin/monerod \
2  --stagenet \
3  --p2p-bind-port 30010 \
4  --rpc-bind-port 30011 \
5  --zmq-rpc-bind-port 30012 \
6  --no-igd \
7  --hide-my-port \
8  --log-level 0 \
9  --data-dir ~/Monero/stagenet/node_01 \
10 --add-exclusive-node 127.0.0.1:30020 \
11 --add-exclusive-node 127.0.0.1:30030 \
12 --add-exclusive-node 127.0.0.1:30040 \
13 --fixed-difficulty 100 \
14 --rpc-bind-ip 0.0.0.0 \
15 --p2p-bind-ip 127.0.0.1 \
16 --confirm-external-bind
17
```

Figura 13: Esempio di script per l'esecuzione di un nodo

In seguito è stato compilato V-Monero, come descritto nel paragrafo 5.3.1.

Successivamente sono stati copiati degli script utilizzati per lanciare l'esecuzione della blockchain ed interagire con essa. Si osservi che dopo la copia dei file è stato necessario renderli eseguibili, cambiando i permessi:

```
chmod +x ./*
```

Nella figura 13 viene mostrato un esempio di script per lanciare l'esecuzione di un nodo. In esso sono presenti diversi parametri che specificano le caratteristiche della rete. I principali sono:

- stagenet: il tipo di rete;
- log-level: il livello di dettaglio dei messaggi visualizzati su una scala da zero a cinque. Il valore zero mostra un numero di messaggi limitato;
- data-dir, cioè la directory contenente i dati della blockchain;
- add-exclusive-node: specifica l'indirizzo e la porta di un altro nodo. In questo modo è possibile creare una rete privata, elencando i diversi nodi coinvolti;
- hide-my-port: la porta non è visibile ad un nodo esterno alla rete;

- *fixed-difficulty*: anzichè incrementare la difficoltà della PoW con il passare del tempo, l'algoritmo di consenso applica una difficoltà fissata. Questa scelta viene fatta con il solo scopo di evitare uno spreco di risorse computazionali durante l'esperimento.

Poi è stato installato MongoDB seguendo le indicazioni della documentazione ufficiale, disponibile all'indirizzo <https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/>.

Infine è stato installato Node Js e la rispettiva versione del Node package manager (NPM). Per fare ciò è stato necessario installare prima curl tramite il comando

```
sudo apt install curl
```

Poi è stata installata la versione LTS (Long Term Support) di Node Js con i comandi:

```
curl -sL https://deb.nodesource.com/setup_12.x | sudo bash -  
sudo apt install nodejs
```

Prima di eseguire il server, bisogna lanciare l'esecuzione di MongoDB e della blockchain. Per eseguire MongoDB, si utilizza il comando

```
sudo service mongod start
```

Per lanciare l'esecuzione di V-Monero bisogna recarsi nella directory `/Scrivania/code/script` ed eseguire tramite Screen (quindi su terminali virtuali differenti) gli script `node01.sh`, `node02.sh`, `node03.sh`, `node04.sh` e `start-server-rpc.sh`.

Affinché il sistema di voto funzioni correttamente, bisogna copiare il wallet amministratore nel percorso `/Monero/stagenet/wallets`. Per fare ciò si usa il comando

```
cp ~/Scrivania/code/script/temp_account/amministratore* ~/Monero/stagenet/  
wallets
```

Il server di V-Monero è disponibile all'indirizzo <https://github.com/Luca-Virgilio/v-monero-be>. Il progetto è strutturato secondo il pattern Model View Controller (MVC). La cartella *public* contiene il codice javascript e le pagine html utilizzate come frontend. Per la realizzazione del frontend è stato utilizzato Bootstrap, un *toolkit* che permette di creare pagine web *responsive* e *mobile-first*.

La cartella *bin* contiene il codice eseguito prima dell'avvio del server. La cartella *app_server* contiene il file `pdkdf2.js` che gestisce la parte crittografica. La funzione di hash utilizzata è SHA-512. In questo file viene creata e salvata nel database una stringa di 16 byte impiegata tutte le volte che viene chiamata la funzione di hash. La cartella *app_api* è a sua volta organizzata in diverse cartelle. *Routers* contiene la logica di instradamento delle diverse API, *models* raccoglie i file che definiscono le collezioni presenti in MongoDB, mentre la cartella *controller* racchiude la logica del comportamento del server.

Infine all'interno della cartella *lib*, nel file *blockchain.js* sono definite tutte le funzioni che interagiscono con il server RPC.

Prima di lanciare l'esecuzione del server bisogna installare le librerie, tramite il comando

```
npm install
```

Per eseguire il server basta digitare nella shell

```
npm run start
```

Quando viene lanciato il server, come prima cosa viene controllata l'esistenza del wallet appartenente all'amministratore. Nel caso in cui non venga trovato, il programma termina. Superato il controllo dell'amministratore, il flusso procede con il controllo dei wallet dei candidati. Nel caso in cui non siano presenti, vengono generati e salvati nel database. Nel database *voting*, esistono quattro diverse collezioni:

- *info*: contiene un documento che specifica se la votazione è in corso;
- *salt*: comprende un documento che memorizza il valore casuale utilizzato nelle funzioni di hash;
- *user*: per ogni utente che partecipa alla votazione viene aggiunto un identificativo anonimo generato in modo deterministico;
- *wallet*: racchiude le informazioni inerenti a tutti i wallet utilizzati. Viene specificato il tipo di wallet, cioè se appartenente ad un elettore o ad un candidato, il nome, l'indirizzo e il valore del saldo del wallet (utilizzato solo nel caso dei wallet candidati al termine di una votazione, in modo che ogni volta che venga richiesto di visualizzare i risultati non si interagisca con la blockchain). Inoltre esistono altri due attributi booleani. L'attributo *loaded* determina se il wallet ha ricevuto i token dall'amministratore, mentre *isUsed* stabilisce se il wallet è già stato utilizzato per un voto.

Dopo aver verificato i wallet dei candidati, si procede alla verifica del numero di wallet utilizzabili dagli elettori. Ogni elettore utilizza un wallet che non potrà più essere utilizzato. Per questo esperimento si è scelto di creare 200 wallet di tipo elettore. Successivamente, prima di procedere si attende che l'amministratore abbia effettuato il mining di 100 blocchi. Questo numero è indicativo ma necessario, poiché incrementa la lunghezza della blockchain e soprattutto il wallet dell'amministratore raggiunge un saldo spendibile, sufficiente per distribuire i token agli elettori. Si ricorda che nella blockchain di Monero per spendere gli output provenienti da una coinbase transaction bisogna attendere 60 conferme del blocco. Quindi qualora si decida di utilizzare un numero maggiore di elettori bisogna verificare che il saldo a disposizione dell'amministratore sia sufficiente.



Figura 14: Homepage di V-Monero

Si osservi che in questa fase il server non è ancora disponibile per effettuare alcun voto.

Ora che l'amministratore dispone di una quantità di token sufficienti, invia i token agli elettori. Visto che in questo esperimento non è presente alcuna fase di registrazione, gli indirizzi dei wallet degli elettori vengono recuperati dal database. Per una maggiore uniformità delle transazioni, il numero di output di una transazione dell'amministratore dovrebbe essere pari al numero di output di una transazione di un qualsiasi elettore, ovvero il numero di candidati più uno. A causa delle modifiche apportate al protocollo V-Monero, per favorire una maggiore trasparenza, non è difficile distinguere quali sono le transazioni appartenenti all'amministratore. Per questo motivo si è scelto come numero indicativo di output 12, ovvero 11 elettori più l'indirizzo per il resto. Questa scelta permette di svolgere più velocemente la distribuzione dei token.

Infine l'ultima azione svolta nella fase di setup è impostare il periodo di mining. Ogni mezz'ora il wallet amministratore effettua il mining per 20 secondi. Si osservi che durante l'operazione di mining il server è bloccato e non è possibile effettuare alcuna operazione di voto. Grazie alla bassa difficoltà fissata, in questo lasso di tempo vengono validati poco meno di una decina di blocchi.

Terminata la fase di setup, il server è disponibile ed è possibile effettuare il voto. L'homepage del sito è visibile nella figura 14. Per partecipare alla votazione è necessario inserire un codice fiscale e confermare. Questa pratica viene utilizzata come fase di autenticazione sebbene non sia possibile verificare l'autenticità del codice fiscale o che

The screenshot shows a web interface for an electronic voting system. At the top, there is a blue header with the logo of the University of Milan-Bicocca and the text 'Università degli studi di Milano-Bicocca'. Below the header, the title 'Sistema di voto elettronico basato su blockchain' is displayed. The main section is titled 'Descrizione' and contains the instruction: 'Inserire il codice fiscale per esprimere il proprio voto. Sarà possibile partecipare una sola volta alla votazione'. There is a text input field containing the tax code 'DLSFMZ80B08E694'. Below the input field, a red error message reads 'Inserire un codice fiscale valido'. To the right of the input field is a blue button labeled 'Avanti'. Below the input field and the error message is a grey button labeled 'Verifica transaction Id'. At the bottom left, there is a small copyright notice: '© v-monero test, Luca Virgilio, 2020'.

Figura 15: Errore in caso di codice fiscale errato

il proprietario corrisponda effettivamente all'utente. Il server controlla solo la sintassi del codice fiscale. Se il codice fiscale è corretto, viene salvato nel session storage e si viene reindirizzati alla pagina successiva. Al contrario, nel caso in cui il codice fiscale sia errato oppure sia già stato utilizzato, viene notificato un errore a schermo, come si evince dalle figure 15 e 16.

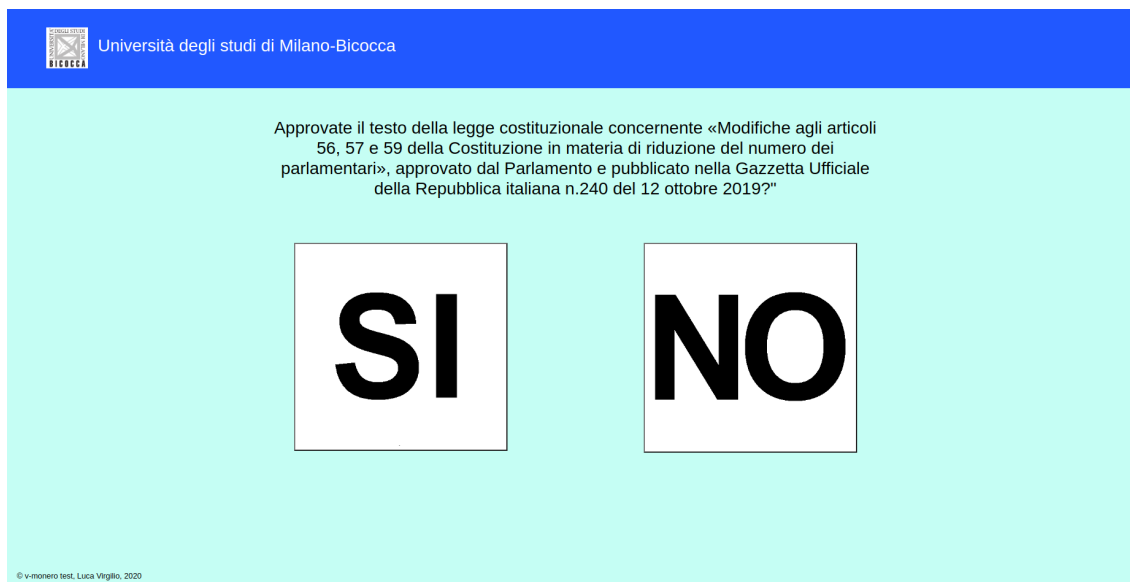
Nella seconda pagina si sottopone il quesito a cui è possibile votare effettuando un click su sì o no (figura 17). Effettuando il voto, viene inviata al backend una richiesta POST contenente la propria scelta e il codice fiscale utilizzato. Prima di effettuare la transazione, il server ricontrolla il codice fiscale. Superato questo controllo, ottiene dal database le informazioni di un wallet elettore non ancora utilizzato per la votazione e procede con la transazione. Nel caso in cui si generi un qualche tipo di errore, esso viene visualizzato a schermo. Quando la transazione ha esito positivo, cioè viene inserita nel mining pool, si genera un identificativo univoco anonimo per l'utente, grazie alla funzione di hash che utilizza come input il codice fiscale e il random salt. Lato utente si viene reindirizzati alla pagina dei ringraziamenti.

Nella pagina dei ringraziamenti, figura 18, viene mostrato il proprio transaction Id, ovvero l'hash della transazione appena effettuata. Copiandolo con un semplice click, nella pagina principale è possibile verificare che la transazione sia stata validata, ovve-



The screenshot shows the 'Sistema di voto elettronico basato su blockchain' interface. At the top, the logo of the Università degli studi di Milano-Bicocca is visible. The main heading is 'Sistema di voto elettronico basato su blockchain'. Below it, the section 'Descrizione' contains the instruction: 'Inserire il codice fiscale per esprimere il proprio voto. Sarà possibile partecipare una sola volta alla votazione'. A text input field contains the fiscal code 'KZOKVC69A54Z354P'. Below the input field, a red error message states: 'L'utente con questo codice fiscale ha già espresso il voto'. To the right of the input field is a blue button labeled 'Avanti'. Below the error message is a grey button labeled 'Verifica transaction Id'. At the bottom left, the copyright notice '© v-monero test, Luca Virgilio, 2020' is present.

Figura 16: Errore in caso di codice fiscale già utilizzato



The screenshot shows the same interface as Figure 16, but with a different question. The question text is: 'Approvate il testo della legge costituzionale concernente «Modifiche agli articoli 56, 57 e 59 della Costituzione in materia di riduzione del numero dei parlamentari», approvato dal Parlamento e pubblicato nella Gazzetta Ufficiale della Repubblica italiana n.240 del 12 ottobre 2019?'. Below the question are two large white boxes with black borders, containing the words 'SI' and 'NO' respectively. At the bottom left, the copyright notice '© v-monero test, Luca Virgilio, 2020' is present.

Figura 17: Pagina contenente il quesito sottoposto



Figura 18: Pagina dei ringraziamenti

ro che sia stato minato un blocco contenente la transazione con quell'identificativo. Più precisamente, nella pagina principale quando si effettua un click sul bottone verifica transaction Id, si apre una finestra in cui è possibile inserire l'hash della transazione (figura 19). Per una maggiore affidabilità, nel caso in cui la transazione non sia stata validata in un blocco, l'errore chiede di attendere un'ora dal momento in cui si è effettuato il voto (figura 20).

Per terminare una votazione, è necessario cambiare il parametro isVoting nella collection info all'interno di MongoDB. Inoltre, spostandosi nel branch politicalElection-end del repository, viene modificata la pagina di homepage, come mostrato in figura 21. A questo punto non è più possibile partecipare alla votazione. Si può verificare un transaction Id oppure vedere i risultati cliccando sull'apposito bottone. I risultati vengono visualizzati tramite un istogramma (figura 22).



Figura 19: Verifica del transaction Id



Figura 20: Errore nella verifica del transaction Id



Figura 21: Homepage al termine della votazione



Figura 22: Pagina dei risultati



Figura 23: Risultati della votazione effettuata con V-Monero

6 Risultati ottenuti

Per la sperimentazione si è scelto effettuare una votazione ombra, riproponendo il quesito posto agli italiani nel referendum del 20 e 21 settembre 2020, riguardo la riduzione del numero di parlamentari.

È stato possibile votare dal 23 settembre alle 12:00 fino al 25 settembre alle 12:00. Il sistema è rimasto in funzione fino a sabato 26, dando la possibilità di verificare i transaction Id e di visualizzare i risultati della votazione. L'esperimento è stato pubblicizzato su Facebook e Whatsapp. Come mostrato in figura 23, alla votazione hanno partecipato 57 persone, con una leggera preferenza per il sì. Non è interesse di questa tesi l'attinenza dei risultati o il campione statistico che ha partecipato alla votazione.

Nella prima parte del capitolo vengono analizzate le proprietà di sicurezza della blockchain e la validità dei requisiti elencati nella tabella 3. Successivamente vengono discusse le criticità e i problemi che affliggono il sistema di voto.

6.1 Analisi di sicurezza

Al termine dell'esperimento, per prima cosa si è verificato che la somma dei voti registrati nei wallet dei candidati fosse pari al numero di elettori che hanno partecipato alla votazione. Nel database sono stati salvati 57 identificativi. Questa è la prova che i voti non sono stati falsificati.

Per verificare la correttezza del protocollo, è stato creato un apposito programma. Il suddetto programma ha il compito di scansionare l'intera blockchain e verificare la corretta composizione delle ring signature. Il codice di questo programma è disponibile nella repository gitHub all'indirizzo <https://github.com/Luca-Virgilio/testMonero>. Prima di tutto bisogna clonare la repository. All'interno del progetto bisogna creare una cartella build:

```
mkdir build && cd build
```

Successivamente bisogna creare il makefile tramite il comando:

```
cmake -DMONERO_DIR=/path/to/monero_folder ..
```

Prima di compilare il progetto bisogna installare la libreria pcsc-lite

```
sudo apt-get update  
sudo apt-get install libpcsc-lite-dev
```

Infine bisogna accedere alla cartella build e compilare il progetto:

```
make
```

Per assicurarsi il corretto funzionamento del programma, bisogna tener presente che all'interno del file main.cpp viene specificato il tipo di network, il percorso in cui è presente una copia della blockchain e l'indirizzo dell'amministratore e la sua private view key.

Durante la votazione sono stati minati 1195 blocchi. Tutte le ring signature realizzate risultano correttamente generate, secondo quanto espresso nel paragrafo 5.3.2 . Utilizzando un wallet elettore, anzichè trasferire il proprio token all'indirizzo di un candidato si è scelto di trasferirlo ad un altro elettore. La transazione è stata completata con successo. Successivamente, quando si è provato ad utilizzare questo token, il sistema lo ha riconosciuto come non spendibile e, come viene mostrato in figura 24, la transazione è stata rifiutata.

Si è verificata la possibilità di spendere una coinbase transaction da parte di un wallet elettore che ha effettuato il mining. Dopo aver atteso 60 blocchi di conferma, si è provato a spendere un output prodotto dalla coinbase transaction. Anche in questo caso, il server RPC ha rifiutato la transazione, notificando lo stesso tipo di errore.

In questo modo è stato dimostrato che chiunque ha accesso alla rete può minare i blocchi ma le coinbase transaction non sono spendibili. Oltre alle transazioni prodotte dall'amministratore, le uniche transazioni permesse sono quelle degli elettori, a condizione che il token speso non sia mai stato trasferito.

Per effettuare ulteriori analisi, si è deciso di utilizzare un progetto open source presente nella repository gitHub <https://github.com/moneroexamples/transactions-export> .

```

2020-09-25 15:47:33.800 0 Starting with 19 non-dust outputs and 0 dust outputs
2020-09-25 15:47:33.800 0 checking preferred
2020-09-25 15:47:33.800 0 estimated bulletproof rct tx size for 2 inputs with ring size 11 and 2 outputs: 2565 (1536 saved)
2020-09-25 15:47:33.800 0 pick_preferred_rct_inputs: needed_money 5.00000130000
2020-09-25 15:47:33.800 0 We can use 2.10e+01 35.158946995978
2020-09-25 15:47:33.800 1 Found preferred rct inputs for rct tx: 2 (35.158946995978)
2020-09-25 15:47:33.800 0 done checking preferred
2020-09-25 15:47:33.800 0 Start of loop with 19 0, tx.dsts.size() 0
2020-09-25 15:47:33.800 0 unused_transfers_idxCses: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
2020-09-25 15:47:33.800 0 unused_dust_idxCses:
2020-09-25 15:47:33.800 0 dsts size 1, first 5.000000000000
2020-09-25 15:47:33.800 0 adding_fee 0, use_rct 1
2020-09-25 15:47:33.800 0 Picking output 2, amount 35.158946995978, kl <=0bfeeae083ae4115bc8a518e52a7392a9462fcbdd42dfdb536c14ebdaf74d>
2020-09-25 15:47:33.800 0 estimated bulletproof rct tx size for 1 inputs with ring size 11 and 1 outputs: 1594 (760 saved)
2020-09-25 15:47:33.800 0 We can fully pay 5AAWHMS33XJL5XE524m99Z83wXyWfCRvJfRd39QhUEKX2Cblat1fCUs1XhuKqS0372ZaTMBRfCwShgntURhF1s for 5.000000000000
2020-09-25 15:47:33.800 0 Considering whether to create a tx now, 1 inputs, tx limit 149400
2020-09-25 15:47:33.800 0 estimated bulletproof rct tx size for 1 inputs with ring size 11 and 2 outputs: 1736 (800 saved)
2020-09-25 15:47:33.800 0 estimated bulletproof rct tx size for 1 inputs with ring size 11 and 2 outputs: 1736 (800 saved)
2020-09-25 15:47:33.800 0 Trying to create a tx now, with 1 outputs and 1 inputs
2020-09-25 15:47:33.800 0 using v5 rules
2020-09-25 15:47:33.800 0 using v0 rules
2020-09-25 15:47:33.800 0 transfer_selected_rct: starting with fee 0.000406850000
2020-09-25 15:47:33.800 0 selected transfers: 2
2020-09-25 15:47:33.800 0 transfer: adding 5.000000000000, for a total of 5.000406850000
2020-09-25 15:47:33.800 0 wanted 5.000406850000, found 35.158946995978, fee 0.000406850000
2020-09-25 15:47:33.800 0 fake outputs: 10
2020-09-25 15:47:33.801 0 Osmon is recent enough, requesting rct distribution
2020-09-25 15:47:33.801 0 account address: 58UlnovJ7zTMczzrj3ZnUPB5E23zK4FK6zph9vGcAdPFMH9S70LQDLQ7XvqNzAGoVHRZ937UCShsarrCct32r5zFN
2020-09-25 15:47:33.836 0 voting global index: 50
2020-09-25 15:47:33.936 0 base_requested_outputs_count: 17
2020-09-25 15:47:33.936 1 554 unlocked rct outputs
2020-09-25 15:47:33.936 1 Fake output mkaups: 67 requested: 0 recent, 0 pre-fork, 0 post-fork, 67 full-chain
2020-09-25 15:47:33.936 0 asking for outputs with amount 0.000000000000: 117 118 119 120 121 122 123 124 125 126 127 134 135 136 137 138 139 140 141 142 144 145 150 152 153 154 155 156 157 158 159 160 161
164 165 166 167 168 169 170 192 193 195 196 197 198 199 200 201 212
2020-09-25 15:47:33.937 1 Error: out of funds. Trying to get more from internal wallet
2020-09-25 15:47:33.937 W /home/luca/src/vanitea/code/vanitea/src/wallet/wallet2.cpp:8259: NtsoolsError21walet_internal_error: Requested real output is not a spendable token
2020-09-25 15:47:33.943 0 do_send_chunk() NOW SENDS: packet=294 B
2020-09-25 15:47:33.943 0 Destructing connection #0 to 0.0.0.0
2020-09-25 15:47:33.956 0 handle_accept
2020-09-25 15:47:33.956 0 Hand server for RPC connections, SSL autodetection
2020-09-25 15:47:33.956 0 set m_connection_type = RPC
2020-09-25 15:47:33.956 0 Spawned connection #8 to 0.0.0.0 currently we have sockets count:2
2020-09-25 15:47:33.956 0 test, connection constructor set m_connection_type=1
2020-09-25 15:47:33.956 0 connection type RPC 127.0.0.1:30014 <==> 127.0.0.1:33048 (vls 127.0.0.1:33048)
2020-09-25 15:47:33.956 0 SSL detection buffer, 190 bytes: 80 79 83 84 32 47 106 115 111
2020-09-25 15:47:33.956 0 That does not look like SSL
2020-09-25 15:47:33.956 1 HTTP [127.0.0.1] POST /json_rpc
2020-09-25 15:47:33.956 1 [127.0.0.1:33048] Incl Calling RPC method close_wallet
2020-09-25 15:47:33.956 0 Problems at ssl shutdown: uninitialized
2020-09-25 15:47:33.956 0 Problems at cancel: Bad file descriptor
2020-09-25 15:47:33.956 0 Problems at shutdown: Bad file descriptor
2020-09-25 15:47:33.957 0 /json_rpc[close_wallet] processed with 0s/0ms
2020-09-25 15:47:33.957 0 do_send_chunk() NOW SENDS: packet=218 B
2020-09-25 15:47:33.957 0 destructing connection #7 to 0.0.0.0

```

Figura 24: Errore generato da una transazione che utilizza un output non spendibile

Questo software genera dei file csv in cui vengono riportate delle analisi sulle transazioni di uno specifico wallet. Prima di compilare il progetto è stato necessario modificare il file `tool.sh` poichè il codice che analizzava la ring signature non era aggiornato alla versione 14 di Monero. Quindi si è utilizzato lo stesso codice presente nella libreria di `testMonero` (recuperato dal codice presente in Monero). Le istruzioni per la compilazione di questa repository sono le stesse di `TestMonero`.

Per eseguire l'analisi di un determinato wallet, bisogna passare come parametri il percorso in cui è presente una copia della blockchain, il tipo di rete che si vuole utilizzare, l'indirizzo e la private view key del wallet. Ad esempio, per ispezionare le transazioni dell'amministratore si è utilizzato

```
./xmr2csv -m -b ~/Monero/stagenet/node_01/stagenet/lmdb --stagenet
-a 58EnGZ7JPUyTANYyZWizYfZoAjnVuWvP3FsMCF74UJHVsMnT3dkHfuFCucwK5xJ7MKAi
  zex3RhA3DnYTXRw7rFw6vFniAU
-v 8a12073d5292a8e99a7eec4f70c5d24c2cef3290121e61b9fd916ed0461dd300
```

Sono state analizzate le transazioni del wallet amministratore e degli elettori numero 9, 10, 33, 48. I wallet elettori sono stati scelti a caso. Si ricorda che l'amministratore ha gestito le operazioni di mining, quindi ha generato molte coinbase transaction. Invece gli elettori hanno ricevuto un output proveniente dalla transazione generata dell'amministratore, e uno come resto nella transazione effettuata per esprimere il proprio voto. Le transazioni appartenenti all'amministratore non mostrano alcun dato inusuale. Al

Nome wallet	Frequenza output nelle ring signature
testEle9	2
testEle10	2
testEle33	3
testEle48	1

Tabella 4: Frequenze degli output appartenenti ai wallet analizzati

contrario l'analisi sugli output appartenenti agli elettori rivela dei risultati inaspettati. Infatti gli output degli elettori sono stati coinvolti in un esiguo numero di ring signature. L'output appartenente all'elettore numero 33 è stato incluso in 3 ring signature mentre quello del numero 48 in una sola transazione. Dalle sole informazioni presenti nella blockchain non è possibile identificare il mittente della transazione, però questo comportamento potrebbe potenzialmente minare l'anonimato di questa e di altre transazioni. Il meccanismo della ring signature si indebolisce in quante più transazioni risulta chiaro l'indirizzo del mittente.

In generale il numero di ring signature che coinvolgono gli output analizzati risulta essere troppo basso. Infatti grazie all'utilizzo di metadati acquisibili dai software wallet si possono produrre euristiche che identificano i mittenti delle transazioni.

La bassa frequenza degli output nelle ring signature è dovuta al fatto che solo 57 persone su 200 wallet disponibili hanno effettuato un voto. Bisogna studiare quanto l'affluenza al sistema di voto elettronico possa incidere sul meccanismo della ring signature. Sebbene sia stato analizzato un campione piccolo di transazioni generate dagli elettori, è necessario approfondire questa potenziale problematica tramite ulteriori esperimenti ed analisi.

Ora si valutano le proprietà elencate nella tabella 3.

Per quanto concerne la proprietà di autenticità, essa non può essere garantita. Durante l'esperimento effettuato, per verificare l'identità degli utenti ed assicurarsi che non abbiano già espresso un voto, si è chiesto di inserire il codice fiscale. Come spiegato nel paragrafo 5.3.4, non c'è garanzia che il codice fiscale sia autentico o che corrisponda a quello dell'utente. In generale nel protocollo di voto, solo chi possiede le chiavi del wallet può effettuare la transazione. Per garantire la proprietà di autenticità si suppone che esse non siano cedibili. Non è stato possibile trovare una strategia che garantisca la non trasferibilità delle chiavi, poiché questo meccanismo deve essere sviluppato sul dispositivo che esegue l'applicazione per votare. Le chiavi devono essere salvate in un'allocazione di memoria cifrata, e l'accesso deve essere limitato tramite credenziali o dati biometrici. Questa considerazione non è comunque sufficiente a garantire l'autenticità del sistema. Infatti nella fase di registrazione l'elettore deve comunicare all'organizzazione l'indirizzo del proprio wallet. Supponendo che utilizzi un sistema

Requisiti	Valutazione
Autenticità	Il problema dell'autenticazione degli elettori rimane una questione aperta
Singolarità	Il comportamento dell'amministratore nel rilasciare i token agli elettori è trasparente e verificabile. Non è possibile effettuare attacchi di doppia spesa
Anonimato	Tramite ring signature, stealth address e bulletproof le transazioni di Monero risultano anonime
Integrità/immutabilità	Garantita grazie agli hash dei blocchi, alla PoW ed eventualmente alla notarizzazione
Non coercibilità	Problematica ancora aperta
Verificabilità	Si può verificare che una transazione sia andata a buon fine. Inoltre è possibile controllare il comportamento generale del sistema
Auditabilità / certificabilità	Il sistema di voto è open source e può essere verificato e certificato da operatori esterni. Al termine della votazione si effettua una notarizzazione su una blockchain pubblica
Mobilità	Il sistema di voto risulta accessibile da qualsiasi luogo
Trasparenza	Ogni nodo della rete possiede una copia locale della blockchain. Serve una campagna di sensibilizzazione per spiegare il funzionamento del sistema agli elettori
Disponibilità	Il sistema risulta sempre disponibile grazie all'utilizzo di AWS
Accessibilità	Favorita da device personali e personalizzabili
Capacità di controllo /ripristino	Protocollo rigido e sistema decentralizzato

Tabella 5: Riassunto delle modalità con cui V-Monero rispetta le proprietà di un sistema di voto

gestito dall'organizzatore delle elezioni, anche le credenziali d'accesso a quell'applicazione web sono parte del processo di autenticazione. Inoltre bisognerebbe garantire che l'indirizzo inserito sia realmente posseduto dall'utente che lo comunica.

La proprietà di singolarità è garantita da diversi fattori. In primo luogo il codice che si occupa di assegnare agli elettori un solo token può essere verificato e ispezionato da operatori esterni, oppure essere reso pubblico. In ogni caso sono note alcune informazioni che possono far trasparire la correttezza di una transazione. Infatti l'amministratore utilizza una transazione coinbase di cui è noto il valore. Tra i diversi output generati, uno di essi corrisponde al resto. Dato che la private view key dell'amministratore è nota a tutti, è possibile conoscere l'ammontare di questa transazione e dedurre dal numero degli altri output se la transazione è stata generata correttamente. In secondo luogo il numero totale di output spendibili prodotti dall'amministratore deve essere pari al numero di elettori che partecipano all'elezione tramite il sistema di voto elettronico. Infine Monero, tramite la key image, evita il problema della doppia spesa, impedendo che un output possa essere speso più volte.

Come descritto nella sezione 5.1, la blockchain di Monero è una blockchain particolarmente attenta all'anonimato. Tramite il meccanismo della ring signature viene nascosto il mittente della transazione, con l'utilizzo dello stealth address vengono celati i destinatari e per mezzo della bulletproof viene provato che la quantità spesa è un valore positivo non superiore agli output utilizzati come input della transazione.

L'integrità e l'immutabilità sono caratteristiche fondamentali della blockchain, nonché due dei valori cardine. La proprietà di integrità viene garantita dall'hash del blocco che viene incluso nel blocco successivo. L'immutabilità della blockchain viene garantita dalla PoW. Infatti la potenza computazionale necessaria per riscrivere l'intera blockchain è estremamente elevata, considerando che più la catena è lunga e più è sicura. In Monero si considera consolidata una transazione che ha ricevuto 10 blocchi di conferma. Dato che in una votazione la catena potrebbe non essere molto lunga, la notarizzazione su un'altra blockchain pubblica è un ulteriore fattore per assicurare l'integrità e l'immutabilità della blockchain.

La proprietà di non coercibilità è la problematica principale insieme all'autenticazione. Rispetto ad un sistema basato su carta, è più difficile assicurare la non coercibilità poiché un elettore può votare da qualsiasi luogo. Essa viene discussa nel prossimo paragrafo.

In un sistema da remoto, la verificabilità entra in conflitto con la proprietà di non coercibilità. Infatti se uno può dimostrare a chiunque la scelta espressa, esso è soggetto a coercizione. Per questo motivo è necessario trovare un equilibrio. In V-Monero è possibile verificare che il proprio voto sia stato incluso in un blocco e quindi conteggiato,

tramite l'utilizzo del transaction Id. Nel protocollo di voto sono previsti dei blockchain explorer tramite i quali è possibile cercare i transaction Id. Inoltre, scansionando la blockchain è possibile rintracciare le transazioni appartenenti all'amministratore e verificare che la somma dei voti ricevuti dai candidati non sia superiore al numero di voti espressi dagli elettori.

Affinché il sistema di voto sia testato, il codice può essere reso pubblico oppure si possono utilizzare operatori esterni per verificarne la sicurezza e l'affidabilità. Per quanto concerne la certificabilità, al termine della votazione l'hash della blockchain di V-Monero viene notarizzata su una blockchain pubblica.

In quanto sistema di voto da remoto, la proprietà di mobilità viene garantita a condizione che sia disponibile una connessione ad Internet.

La proprietà di trasparenza è una caratteristica delle blockchain pubbliche. Ogni nodo della rete possiede una copia in locale dell'intera blockchain. Per quanto riguarda la percezione dell'elettore, è sicuramente necessaria una campagna di sensibilizzazione in ambito digitale, con particolare attenzione ad argomenti come la blockchain e la sicurezza informatica.

Per quanto riguarda l'esperimento effettuato, il sistema di voto è stato disponibile per il 98% del tempo. La mancanza di disponibilità al 100% è dovuta al fatto che il server doveva occuparsi di gestire le operazioni di mining. Infatti ogni mezz'ora il wallet admin eseguiva per 20 secondi il mining. Questa è una limitazione solo dell'esperimento effettuato. Infatti è possibile gestire le operazioni di mining tramite un'altro programma, oppure far sì che vengano gestite in maniera indipendente da altri nodi. In generale grazie all'utilizzo di AWS il sistema di voto è sempre disponibile. Per ottenere una migliore affidabilità del sistema è possibile utilizzare nodi eseguiti in regioni AWS diverse.

Il sistema di voto è accessibile a qualsiasi tipo di persona che dispone di un dispositivo con una connessione a Internet. Apparentemente questa può sembrare una limitazione ma oggi giorno uno smartphone o un PC sono indispensabili. Quindi è possibile supporre che tutti ne possiedano almeno uno. Il vantaggio dell'utilizzo di un device personale è la possibilità di sfruttare caratteristiche già presenti che facilitano l'accessibilità, come ad esempio la modalità "non vedenti" presente sugli smartphone.

Infine, per quanto riguarda la capacità di controllo e ripristino, V-Monero ha un protocollo preciso e ben definito. In caso di comportamenti anomali da parte di un nodo, essendo un sistema decentralizzato non viene inficiata la sicurezza e l'affidabilità. Inoltre tutti i nodi possiedono una copia della blockchain da cui eventualmente effettuare operazioni di ripristino.

6.2 Problemi e criticità

Attualmente esistono diversi problemi e criticità che affliggono questo sistema di voto. Il principale problema, come è stato anticipato nel capitolo precedente, è quello dell'autenticazione. Questo problema è strettamente collegato alla proprietà di non coercizione. Infatti se non esiste un meccanismo che certifichi l'identità di chi sta effettuando il voto, un elettore potrebbe cedere il proprio diritto di voto. D'altro canto la soluzione del problema dell'identità digitale non assicura la proprietà di non coercizione. Essendo una votazione da remoto, attualmente non c'è modo di garantire che il voto espresso sia personale e non sia influenzato da persone o agenti esterni. Questo è il principale limite dei sistemi di voto da remoto. Alcuni sistemi per eludere la coercizione permettono agli elettori di ripetere il proprio voto. Partendo dal presupposto che una votazione potrebbe non prevedere questa possibilità, solitamente chi adotta questo meccanismo non può garantire l'anonimato dell'elettore. Questo è il motivo principale per cui la maggior parte dei sistemi elettorali richiedono lo svolgimento in presenza.

Un altro problema è dato dalla sicurezza dell'applicazione utilizzata dagli elettori. Infatti dato che gli elettori in un sistema di voto da remoto utilizzano il proprio dispositivo, bisogna garantire la sicurezza e l'integrità dell'applicazione. Nell'esperimento non è stato possibile verificare queste proprietà che risultano fondamentali per il corretto funzionamento del sistema, poiché gli elettori hanno utilizzato un'applicazione web. È comunque possibile esplicitare alcune osservazioni. L'integrità dell'applicazione è fondamentale poiché eventuali modifiche al software wallet possono compromettere l'anonimato del votante. Ad esempio, se venisse utilizzato un software wallet che sceglie sempre gli stessi decoy per comporre la ring signature (e questi decoy sono noti), il meccanismo per nascondere il mittente sarebbe compromesso.

Inoltre bisogna garantire che il software wallet non memorizzi le private transaction key delle transazioni, altrimenti con questa informazione un elettore potrebbe decifrare l'ammontare della transazione, rivelando il proprio voto (paragrafo 5.1.2).

Un altro potenziale problema è quello che un virus possa salvare in chiaro informazioni quali mittente, destinatario e ammontare delle transazioni.

Un'altra criticità riguarda la fase di conteggio dei voti. Con l'obiettivo di favorire una maggiore trasparenza, gli organizzatori della votazione potrebbero decidere di rendere pubbliche le private view key dei candidati, in modo che chiunque abbia accesso alla blockchain possa conteggiare i risultati delle elezioni. Si osservi che questa operazione può compromettere la segretezza del voto. Infatti qualora un elettore entri in possesso di queste informazioni, potrebbe dimostrare a chiunque il voto effettuato. Si ricordi che in una transazione Monero l'ammontare degli output è cifrato e solo il possessore di quest'ultimi è in grado di generare la corrispondente chiave privata (paragrafo 5.1.2). Per garantire una maggiore segretezza della votazione, quando un elettore effettua un voto, il token viene inviato al wallet del candidato scelto, mentre a ciascun altro can-

didato viene trasferita una piccola quantità (nell'esperimento, $1 * 10^{-5} XMR$). Questo poiché qualora un agente malevolo recuperi le chiavi di un elettore, senza la private transaction key non può identificare a quale indirizzo è stato inviato il token. Invece grazie alle view key dei candidati, si può decifrare il valore dei diversi output della transazione.

In sintesi, la pubblicazione delle private view key dei candidati potrebbe favorire la coercizione.

Infine bisogna valutare l'efficacia della ring signature. Da quanto emerso nel precedente capitolo, nell'esperimento effettuato gli output degli elettori analizzati vengono scelti poche volte come decoy. La scelta dei decoy e la grandezza della ring signature sono problemi che continuano ad essere studiati e valutati anche nella blockchain di Monero. A causa della bassa affluenza dei partecipanti, pari al 28,5% in media ciascun output doveva essere incluso in 3 transazioni. Servono ulteriori analisi su altri esperimenti che coinvolgono un numero differente di elettori, con una diversa affluenza. Nel caso di votazioni con bassa affluenza, una possibile soluzione potrebbe essere quella di utilizzare una ring signature più grande, oppure creare delle finte transazioni. Quest'ultima proposta potrebbe condizionare la trasparenza del sistema. Un'altra valida alternativa è trovare un nuovo algoritmo di scelta delle firme, pensato per questo specifico scenario.

Un'altra criticità che affligge tutte le reti e le blockchain è il problema dei metadati. Quando un wallet si connette ad un remote node, questo acquisisce informazioni come l'indirizzo IP, la geolocalizzazione, la time zone. Queste informazioni possono essere utilizzate per creare euristiche che individuino i mittenti delle transazioni. Il team di Monero conosce questa problematica ed è un obiettivo dei prossimi aggiornamenti mitigare questa criticità. In ogni caso è consigliato l'utilizzo di VPN.

Considerando che in un sistema di voto devono essere salvate informazioni esterne alla blockchain, come gli indirizzi dei wallet degli elettori, oppure le view key dei wallet dei candidati, è importante mantenere questi dati sensibili protetti e partizionati. L'acquisizione di queste informazioni da parte di un agente malevolo può compromettere l'anonimato della votazione.

6.3 Considerazioni finali

Secondo quanto emerso dalle analisi effettuate sull'esperimento, le modifiche apportate alla blockchain di Monero hanno permesso la creazione di un protocollo di voto affidabile e sicuro. Le proprietà definite ed elencate nella tabella 3, risultano essere rispettate, ad eccezione dell'autenticità e della non coercibilità. Queste proprietà, in un sistema di voto da remoto implicano problematiche come l'identità digitale, una tematica molto dibattuta e complessa.

Sebbene non sia stato possibile verificare la sicurezza e l'integrità dell'applicazione da utilizzare per partecipare alla votazione, sono state spiegate alcune problematiche che

possono inficiare l'anonimato dell'elettore.

L'analisi della blockchain ha evidenziato un'anomalia presente nelle ring signature, nel caso in cui solo una parte degli elettori previsti partecipi effettivamente alla votazione. Malgrado siano necessari ulteriori esperimenti per approfondire la questione, essa non desta particolari preoccupazioni.

In ultima analisi, considerando che il protocollo di Monero continua ad essere sviluppato e migliorato nel tempo e alcuni problemi, come quello dell'identità digitale, devono essere risolti per progredire nello sviluppo di una civiltà improntata sul digitale, il protocollo descritto in questo elaborato risulta essere un buon punto di partenza per la costruzione di un sistema di voto da remoto sicuro e affidabile.

7 Conclusioni

7.1 Valutazioni finali e conclusioni

L'obiettivo di questo elaborato era quello di progettare un sistema di voto elettronico basato su blockchain e dimostrazioni zero knowledge. La blockchain è una delle tecnologie più innovative degli ultimi anni e mira a rivoluzionare i più disparati settori, come quello agroalimentare o finanziario. Infatti grazie alla sua struttura e ai suoi protocolli, i sistemi basati su blockchain sono in grado di garantire sicurezza, immutabilità e trasparenza in una rete decentralizzata. La blockchain di Bitcoin è stata la prima a risolvere il problema del consenso distribuito in un sistema decentralizzato.

Malgrado le diverse sperimentazioni in atto in alcuni Stati, non esiste un sistema di voto elettronico riconosciuto universalmente come sicuro e affidabile. Per questo motivo si è deciso di utilizzare la blockchain per progettare un protocollo di voto elettronico. Nel capitolo 3 sono stati descritti i principali sistemi di voto elettronico che utilizzano diversi tipi di blockchain. Diversamente da questi protocolli, si è scelto di valutare l'utilizzo di Zcash e Monero, due blockchain attente all'anonimato. Entrambe le blockchain utilizzano dimostrazioni zero knowledge per garantire l'anonimato delle transazioni. Si è scelto di utilizzare Monero poiché:

- è un protocollo consolidato e in continua evoluzione;
- richiede un minor numero di modifiche per essere adattato in un sistema di voto;
- le dimostrazioni zero knowledge non richiedono alcun trusted setup.

Dato che la blockchain è una tecnologia complessa, si è deciso di progettare un sistema di voto da remoto, utilizzabile nel caso d'uso più complicato, ovvero quello delle elezioni politiche. Il sistema di voto deve assicurare le proprietà di autenticità, anonimità, integrità, disponibilità e garantire la non coercizione.

Nel sistema sono presenti tre attori principali: l'amministratore, gli elettori e i candidati. Nel sistema proposto, esprimere una scelta di voto corrisponde ad effettuare una transazione sulla blockchain.

Per adattare la blockchain di Monero al sistema di voto, sono state effettuate alcune modifiche al protocollo:

- a partire dal secondo blocco, valgono le proprietà della versione 14 di Monero;
- solo i token generati da transazioni provenienti dall'amministratore sono spendibili;
- i token in possesso degli elettori possono essere trasferiti una sola volta. In questo modo non è possibile delegare il proprio voto ad un'altro elettore.

Inoltre i candidati non possono trasferire i propri voti. Durante la fase di spoglio per calcolare i risultati dell'elezione è sufficiente calcolare il saldo dei wallet dei candidati. Per verificare l'affidabilità e la sicurezza del sistema di voto si è scelto di organizzare una votazione online, riproponendo il quesito del referendum sul taglio del numero dei parlamentari. Durante l'esperimento la blockchain è stata gestita da 4 nodi, eseguiti tramite il servizio di AWS.

Analizzando i risultati ottenuti dall'esperimento, si può affermare che tutte le proprietà definite ed elencate nella tabella 3 risultano essere rispettate, ad eccezione dell'autenticità e della non coercibilità. Poichè non esiste ancora un meccanismo che accerti l'identità digitale dell'utilizzatore si è ipotizzato che le chiavi del wallet siano personali e non cedibili. Per quanto riguarda la non coercizione non è possibile garantire la privacy dell'elettore in quando la votazione viene effettuata da remoto con il proprio dispositivo personale. Un ulteriore problema che non è stato approfondito in questa tesi è la sicurezza del dispositivo e dell'applicazione che un elettore utilizza durante la votazione. Invece le proprietà di anonimato, integrità e disponibilità, vengono garantite in questo sistema di voto, dimostrando che è possibile progettare un sistema di voto elettronico basato su blockchain e dimostrazioni zero knowledge.

7.2 Possibili sviluppi futuri

Con l'obiettivo di creare un sistema completo, installabile ed utilizzabile, come prima cosa sarebbe necessaria la creazione di un wallet Monero multiplatforma, modificato secondo quanto espresso nei capitoli precedenti. Successivamente bisognerebbe testare il sistema di voto in diversi scenari, come ad esempio votazioni con un maggior numero di partecipanti e differenti percentuali di affluenza, per verificare l'efficacia della ring signature e assicurarsi che le proprietà del sistema siano sempre garantite. Infine gran parte degli sviluppi futuri dovrebbero concentrarsi sulla risoluzione del problema dell'autenticazione dell'elettore e su un meccanismo per garantire la non coercibilità.

Riferimenti bibliografici

- [1] Referendum autonomia, il voto elettronico in Lombardia funziona così. Maroni è protagonista del tutorial. [La Repubblica](https://milano.repubblica.it/cronaca/2017/10/09/news/referendum_autonomia_lombardia_voto_elettronico_come_funziona-177796742/), 2017. URL https://milano.repubblica.it/cronaca/2017/10/09/news/referendum_autonomia_lombardia_voto_elettronico_come_funziona-177796742/.
- [2] Philip Boucher. How blockchain technology could change our lives: In-depth analysis. 2017. URL https://www.europarl.europa.eu/RegData/etudes/IDAN/2017/581948/EPRS_IDA%282017%29581948_EN.pdf.
- [3] Cresce la Blockchain: 488 progetti nel mondo (+56%) nel 2019 e grandi opportunità per l'Italia, 2020. URL <https://www.osservatori.net/it/ricerche/comunicati-stampa/cresce-la-blockchain-488-progetti-nel-mondo-plus56-nel-2019-e-grandi-opportunita-per-litalia>.
- [4] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, 2008. URL <https://bitcoin.org/bitcoin.pdf>.
- [5] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. [Journal of the ACM \(JACM\)](#), 32(2): 374–382, 1985.
- [6] Vitalik Buterin. On public and private blockchains, 2015. URL <https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/>.
- [7] Lawrence D Norden and Christopher Famighetti. [America's voting machines at risk](#). Brennan Center for Justice at New York University School of Law, 2015.
- [8] D Brown. Standards for efficient cryptography, sec 1: elliptic curve cryptography. [Released Standard Version](#), 1, 2009. URL <https://www.secg.org/sec1-v2.pdf>.
- [9] Nihal R Gowravaram. [Zero Knowledge Proofs and Applications to Financial Regulation](#). PhD thesis, 2018. URL <https://dash.harvard.edu/bitstream/handle/1/38811528/GOWRAVARAM-SENIORTHESIS-2018.pdf?sequence=3&isAllowed=y>.
- [10] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. [SIAM Journal on computing](#), 18(1):186–208, 1989.
- [11] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. [Journal of the ACM \(JACM\)](#), 38(3):690–728, 1991.

- [12] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In **2018 IEEE Symposium on Security and Privacy (SP)**, pages 315–334. IEEE, 2018.
- [13] Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michaël Quisquater, Louis Guillou, Marie Annick Guillou, Gaïd Guillou, Anna Guillou, Gwennolé Guillou, and Soazig Guillou. How to explain zero-knowledge protocols to your children. In **Conference on the Theory and Application of Cryptology**, pages 628–631. Springer, 1989.
- [14] Christian Reitwiessner. zkSNARKs in a nutshell (2016). URL: <http://chriseth.github.io/notes/articles/zksnarks/zksnarks.pdf>, 2019.
- [15] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In **Annual International Conference on the Theory and Applications of Cryptographic Techniques**, pages 626–645. Springer, 2013.
- [16] Sean Bowe, Ariel Gabizon, and Matthew D Green. A multi-party protocol for constructing the public parameters of the pinocchio zk-SNARK. URL <https://eprint.iacr.org/2017/602.pdf>.
- [17] Zcash foundation. Parameter generation. URL <https://z.cash/technology/paramgen/>.
- [18] Benjamin Winston Josh Swihart and Sean Bowe. Zcash counterfeiting vulnerability successfully remediated, 2019. URL <https://electriccoin.co/blog/zcash-counterfeiting-vulnerability-successfully-remediated/>.
- [19] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In **Annual International Conference on the Theory and Applications of Cryptographic Techniques**, pages 327–357. Springer, 2016.
- [20] Tim Ruffing and Giulio Malavolta. Switch commitments: A safety switch for confidential transactions. In **International Conference on Financial Cryptography and Data Security**, pages 170–181. Springer, 2017.
- [21] Kevin Kirby, Anthony Masi, and Fernando Maymi. Votebook. a proposal for a blockchain-based electronic voting system. **The Economist**, 6, 2016.
- [22] Blockchain Technologies Corp. VoteWatcher - The World's Most Transparent Voting Machine., 2016.

- [23] Roman Alyoshkin. Polys online voting system. whitepaper. URL https://polys.me/assets/docs/Polys_whitepaper.pdf.
- [24] Francesco Fusco, Maria Ilaria Lunesu, Filippo Eros Pani, and Andrea Pinna. Crypto-voting, a blockchain based e-voting system. In **10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management**, pages 221–225, 2018.
- [25] Agora. Agora: bringing our voting systems into 21st century. URL <https://www.agora.vote/>.
- [26] Inc Voatz. Voatz mobile voting platform an overview: Security, identity, auditability. URL <https://voatz.com/wp-content/uploads/2020/07/voatz-security-whitepaper.pdf>.
- [27]
- [28] Ryan Osgood. The future of democracy: Blockchain voting. **COMP116: Information security**, pages 1–21, 2016.
- [29] Adi Shamir. How to share a secret. **Communications of the ACM**, 22(11):612–613, 1979.
- [30] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. **IEEE transactions on information theory**, 31(4):469–472, 1985.
- [31] S McMath, F Crabbe, and D Joyner. Continued fractions and parallel squf. **arXiv preprint math/0601263**, 2006.
- [32] Kirill Nikitin, Eleftherios Kokoris-Kogias, Philipp Jovanovic, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Justin Cappos, and Bryan Ford. CHAINIAC: Proactive software-update transparency via collectively signed skipchains and verified builds. In **26th Advanced Computing Systems Association Security Symposium**, pages 1271–1287, 2017.
- [33] Wikipedia. Skip list — wikipedia, the free encyclopedia, 2020. URL https://en.wikipedia.org/wiki/Skip_list.
- [34] Josh Benaloh. Towards simple verifiable elections. In **Proceedings of Workshop on Trustworthy Election (WOTE’06)**, pages 61–68, 2006.
- [35] Dawid Gaweł, Maciej Kosarzecki, Poorvi L Vora, Hua Wu, and Filip Zagórski. Apollo—end-to-end verifiable internet voting with recovery from vote manipulation. In **International Joint Conference on Electronic Voting**, pages 125–143. Springer, 2016.

- [36] C Andrew Neff. A verifiable secret shuffle and its application to e-voting. In **Proceedings of the 8th ACM conference on Computer and Communications Security**, pages 116–125, 2001.
- [37] Valimised. Voting results in detail, 2020. URL <https://rk2019.valimised.ee/en/voting-result/voting-result-main.html>.
- [38] Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, and J Alex Halderman. Security analysis of the estonian internet voting system. In **Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security**, pages 703–715, 2014.
- [39] LLC Proprietary Guardtime Federal. Keyless Signature Infrastructure (KSITM) Technology: An Introduction to KSI Blockchain Technology and Its Benefits. URL http://blockchain.machetmag.com/wp-content/uploads/2017/11/Guardtime_WhitePaper_KSI.pdf, 2017.
- [40] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In **Proceedings of the thirteenth EuroSys conference**, pages 1–15, 2018.
- [41] Michael A Specter, James Koppel, and Daniel Weitzner. The Ballot is Busted Before the Blockchain: A Security Analysis of Voatz, the First Internet Voting Application Used in US Federal Elections. **29th Advanced Computing Systems Association Security Symposium**, 2020.
- [42] Jorge Lopes, J. L. Pereira, and João Varajão. Blockchain Based E-voting System: A Proposal. In **Americas' Conference on Information Systems (AMCIS)**, 2019.
- [43] Fririk Hjálmarsson, Gunnlaugur K Hreiðarsson, Mohammad Hamdaqa, and Gísli Hjálmtýsson. Blockchain-based e-voting system. In **2018 IEEE 11th International Conference on Cloud Computing (CLOUD)**, pages 983–986. IEEE, 2018.
- [44] Monero. Monero. . URL <https://www.getmonero.org/>.
- [45] Monero. Monero. . URL <https://www.getmonero.org/resources/research-lab/>.
- [46] SerHack. **Mastering Monero: The future of private transactions**. 2018. URL <https://github.com/monerobook/monerobook>.
- [47] Monero Community Workgroup. Breaking Monero. URL https://www.youtube.com/watch?v=W0yC60B6ezA&list=PLsSYUeVwrHBNAUre2G_LYDsdo-tD0ov-y.

- [48] Daniel J Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. Twisted edwards curves. In [International Conference on Cryptology in Africa](#), pages 389–405. Springer, 2008.
- [49] Madhura A Patil and Pradeep T Karule. Design and implementation of keccak hash function for cryptography. In [2015 International Conference on Communications and Signal Processing \(ICCSP\)](#), pages 0875–0878. IEEE, 2015.
- [50] Morris J Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions. Technical report, 2015.
- [51] Martin Hellman. An overview of public key cryptography. [IEEE Communications Society Magazine](#), 16(6):24–32, 1978.
- [52] Kurt M Alonso. Zero to Monero. 2020. URL <https://web.getmonero.org/library/Zero-to-Monero-2-0-0.pdf>.
- [53] The Monero Project. OpenAlias. URL <https://openalias.org/>.
- [54] Raffaele Nicodemo Vincenzo "degli Abruzzi" Di Nicola, Calogero Mandracchia. Terminus white paper, a digital voting system for the 21th century. URL https://github.com/theterminusfoundation/terminus/blob/master/terminus_white_paper.pdf.