



UNIVERSITÀ DI PERUGIA  
Dipartimento di Matematica e Informatica



TESI TRIENNALE IN INFORMATICA

# Un sistema di e-voting basato sul protocollo di Tornado Cash

*Relatore*

**Prof. Francesco Santini**

*Laureando*

**Bruno Lazo La Torre Montalvo**

---

Anno Accademico 2021-2022

# Indice

<b>1</b>	<b>Introduzione</b>	<b>4</b>
1.1	Obiettivi . . . . .	4
1.2	Struttura della tesi . . . . .	5
<b>2</b>	<b>Sistemi di votazione</b>	<b>6</b>
2.1	Votazioni mediante schede cartacee . . . . .	6
2.2	Problemi dei sistemi di voto mediante schede cartacee . . . . .	7
2.3	E-voting . . . . .	8
2.4	Proprietà dei sistemi di e-voting . . . . .	8
2.5	Applicazione della blockchain su sistemi di e-voting . . . . .	10
2.6	Applicazioni basate su blockchain presenti in letteratura . . . . .	10
<b>3</b>	<b>Blockchain Ethereum</b>	<b>13</b>
3.1	Blockchain . . . . .	13
3.2	Ethereum . . . . .	14
3.3	Smart contract . . . . .	15
3.4	Solidity . . . . .	16
3.5	dApp . . . . .	17
3.6	Nodi Ethereum . . . . .	17
3.7	Account Ethereum . . . . .	18
3.8	Transazioni Ethereum . . . . .	18
3.9	Algoritmi di consenso: Proof of Work . . . . .	19
3.10	Ether . . . . .	20
3.11	Token ERC20 . . . . .	20
3.12	Network . . . . .	22

<b>4</b>	<b>Tornado Cash</b>	<b>23</b>
4.1	Protocollo decentralizzato e non-custodial . . . . .	23
4.2	Blockchain operative . . . . .	24
4.3	Setup . . . . .	24
4.4	Depositi e ritiri . . . . .	25
4.5	zk-SNARK proof . . . . .	30
4.6	Perdersen Hash . . . . .	31
4.7	Merkle Tree . . . . .	32
4.8	Anonimità . . . . .	33
<b>5</b>	<b>Protocollo di risoluzione proposto</b>	<b>35</b>
5.1	Architettura del progetto . . . . .	37
5.2	Soluzione Obiettivo 1. Distribuzione di un token dedicato alle votazioni	41
5.3	Soluzione Obiettivo 2. Anonimizzazione dell'utente . . . . .	42
5.4	Soluzione Obiettivo 3. Rilascio dApp dedicata alle votazioni . . . . .	50
5.5	Stima dei costi . . . . .	53
<b>6</b>	<b>Conclusioni</b>	<b>55</b>

# Capitolo 1

## Introduzione

Il voto oltre ad essere un diritto politico è riconosciuto anche come un diritto umano [6]. Il diritto di voto è una delle più importanti forme di manifestazione di libertà personale ed espressione democratica. La base di un governo legittimo e democratico è dato da delle elezioni libere, universali e periodiche.

Grazie al diritto di voto i cittadini hanno l'opportunità di intervenire nelle decisioni del governo. È quindi importante che questo diritto venga esercitato in modo libero, segreto e con opportunità di scelta.

Sono stati quindi oggetto di ricerca, lo studio di sistemi di voto alternativi che garantiscano trasparenza, convenienza e affidabilità.

L'*e-voting* è stato proposto come soluzione ai requisiti sopracitati, i classici sistemi di voto elettronico non risolvono però dei problemi presenti anche nel sistema di votazione cartacea, che sono causati dalle implementazioni centralizzate di essi, quali: sicurezza, privacy e integrità. Oltre a renderli vulnerabili ad attacchi informatici che possono compromettere l'esito delle votazioni.

Tecnologie come la *blockchain*, tramite la loro natura decentralizzata e attributi di anonimità e trasparenza, possono venire utilizzate per affrontare e risolvere questi punti deboli dei sistemi di voto.

### 1.1 Obiettivi

La finalità di questo studio è l'implementazione di un sistema di votazioni elettronico basato sulla tecnologia blockchain, che soddisfi quelle che sono le classiche proprietà

di un sistema di e-voting (come descritto in Sezione 2.4).

Per la realizzazione del sistema sono stati individuati i seguenti obiettivi:

1. Distribuzione di un token dedicato alle votazioni:

L'idea è quella di implementare l'intero sistema all'interno della blockchain Ethereum, creando un token che venga distribuito a un utente autorizzato previa identificazione.

2. Anonimizzazione dell'utente:

Vogliamo implementare un servizio *coin mixer* che permetta di effettuare operazioni di deposito e ritiro sul token di voto in maniera decentralizzata e sicura.

3. Rilascio dApp dedicata alle votazioni:

Infine, abbiamo come scopo quello di realizzare una web dApp che permetta all'utente di esprimere il suo voto spendendo il suo token di voto.

## 1.2 Struttura della tesi

La tesi è organizzata come segue. Nel Capitolo 2 vengono descritti i meccanismi e procedure dei sistemi di voto, siano quelli tradizionali, elettronici ed elettronici su blockchain. Analizzando le differenze tra di essi, i vantaggi degli uni rispetto agli altri ed esaminando le applicazioni presenti in letteratura. Le proprietà della blockchain Ethereum, la piattaforma su cui si basa il progetto, sono descritte nel Capitolo 3. Nel Capitolo 4 è descritto il funzionamento e le caratteristiche di Tornado Cash, la cui tecnologia verrà utilizzata in questo studio per anonimizzare il token di voto. Nel Capitolo 5 viene descritto il protocollo di risoluzione che mira a risolvere gli obiettivi proposti dallo studio. Infine nel Capitolo 6 verranno presentate le conclusioni di questo studio.

# Capitolo 2

## Sistemi di votazione

### 2.1 Votazioni mediante schede cartacee

Il voto segreto, così come implementato negli attuali sistemi tradizionali, venne introdotto per la prima volta in Tasmania nel 1856 [29], ed è diventato uno standard nella maggior parte dei paesi democratici a fine del ventesimo secolo.

Il voto segreto sarà caratterizzato dall'uso di schede elettorali, standardizzate e omogenee per tutti i votanti, con le seguenti caratteristiche:

- Vengano stampate a spesa pubblica.
- I nomi di tutti i candidati e partiti compaiano in modo uniforme e strutturato.
- Vengano distribuite solo nei seggi ufficiali.
- Possano essere compilate con la dovuta privacy.

Il processo di voto nelle elezioni, come è stato detto, viene svolto in dei seggi prefissati collocati nel territorio del paese. Perciò le persone per poter esprimere il proprio voto dovranno recarsi fisicamente al seggio di appartenenza e compilare a mano la scheda che gli verrà fornita in loco.

Per garantire la privacy, la compilazione della scheda avviene all'interno delle cabine elettorali, delle stanze o cabine in cui l'accesso è riservato ad una sola persona alla volta, in cui si è al riparo dalla vista altrui.

Successivamente il votante inserirà la propria scheda all'interno di una urna sigillata, all'interno della quale saranno presenti altre schede elettorali. terminate le votazioni gli scrutatori verificheranno la validità delle schede di voto e procederanno con il conteggio dei voti. Una volta concluso il conteggio dei voti verranno resi pubblici i risultati.

## 2.2 Problemi dei sistemi di voto mediante schede cartacee

Delle problematiche presenti all'interno di un sistema di voto tradizionale che usa le schede elettorali cartacee [17] sono:

- *Costo*: elevata spesa derivata dall'uso di attrezzature per le operazioni di voto e impiego di risorse umane (scrutatori, presidenti di seggio...).
- *Tempo*: possono trascorrere giorni dal momento in cui viene terminata l'operazione di voto al momento in cui vengono pubblicati i risultati del conteggio. L'operazione stessa di voto è dispendiosa in termini di tempo, dal momento che solo poche persone alla volta potranno votare, mentre le altre saranno costrette ad aspettare.
- *Ambiguità*: problemi nell'interpretazione di una scheda compilata a mano, magari causati da una calligrafia difficile da comprendere o da una crocetta fuori posto.
- *Errore umano*: la procedura di voto e in particolare quella di conteggio dei voti possono essere soggette ad errori. Difatti spesso capita che i risultati dei molteplici conteggi che vengono effettuati manualmente, differiscano tra loro.
- *Bassa accessibilità*: statisticamente il dover raggiungere fisicamente la sede del seggio elettorale costituisce una delle cause maggiori di astensione di voto. Basti pensare alle problematiche che può causare, il raggiungere il seggio a persone con difficoltà motorie, anziani, elettori all'estero...

L'uso di sistemi di votazione elettronica comporterebbe numerosi vantaggi, tra cui la risoluzione dei problemi sopracitati.

## 2.3 E-voting

L'e-voting o votazione elettronica, fa riferimento all'utilizzo di tecnologie elettroniche e informatiche durante il processo di voto o conteggio di esso.

Il primo uso significativo di un sistema di voto elettronico è avvenuto nel 1964, durante le elezioni presidenziali degli Stati Uniti [38], tramite l'uso di *Ballot marking devices* (BMD).

I BMD sono dei dispositivi elettronici che consentono all'utente votante di registrare il suo voto in una scheda elettorale fisica, solitamente vengono usati in concomitanza di *dispositivi a scansione ottica*. Questo tipo di dispositivo scrutinerà le schede in modo automatico, verificando che non vi sia presente alcun tipo anomalia.

Negli ultimi due decenni i *Direct recording electronic systems* (DRE) sono stati ampiamente utilizzati nei sistemi elettorali di paesi come Brasile, India, Venezuela e Stati Uniti [21].

I DRE sono macchine dotate di pulsanti o schermo touchscreen tramite cui l'individuo votante potrà selezionare il suo voto. Spesso i sistemi DRE vengono accoppiati con dispositivi che stampano su carta l'operazione effettuata in modo tale che il votante possa verificare la corretta esecuzione dell'operazione.

Un altro sistema di voto che viene utilizzato in sistemi elettorali moderni è quello via Internet. Il *voto online* è un sistema che permette di votare da qualsiasi dispositivo elettronico che possa accedere a un browser web e abbia accesso ad una connessione Internet.

## 2.4 Proprietà dei sistemi di e-voting

Affinché un sistema di voto elettronico venga considerato idoneo è necessario che dei requisiti basici vengano rispettati. Tutti i requisiti non possono venire affrontati da un singolo protocollo, quindi a seconda del tipo di elezione coinvolta, si farà uso del tipo di protocollo più adatto [26].

Delle classiche proprietà che si ritrovano nei sistemi di e-voting sono [11]:

- *Unicità*: ad un utente non è permesso votare più di una volta, questo è solitamente garantito dall'utilizzo di un token di voto.



- *Integrità*: non è possibile modificare o eliminare un voto, senza che ciò venga rivelato.
- *Conteggio*: il conto dei voti deve essere semplice affinché i risultati possano venire verificati.
- *Autenticazione*: solo gli utenti che si sono correttamente identificati alle autorità di competenza potranno votare.
- *Confidenzialità*: non è possibile ottenere dei risultati intermedi durante le votazioni, questo per evitare che questo tipo di informazioni influenzino la scelta del votante oppure si cerchi di manipolare la scelta dei votanti in qualche maniera.
- *Attendibilità*: indica che il sistema di voto deve risultare affidabile, perciò i voti che sono stati fatti devono venire registrati correttamente.
- *Verificabilità*: possibilità per ciascun votante di accertare che il suo voto sia stato registrato (verificabilità individuale) e che il conteggio dei voti finale sia stato eseguito in modo corretto (verificabilità universale).
- *Privacy*: non è possibile determinare da parte di nessuna entità il voto di un utente.
- *Assenza di evidenza*: gli utenti non devono essere in grado di dimostrare per chi hanno votato, questa caratteristica serve per prevenire la vendita del voto o di venire forzati nella scelta del voto.

Per quanto riguarda queste ultime tre proprietà è stato dimostrato [18] che non è possibile per un sistema di voto soddisfare:

- verificabilità universale e privacy contemporaneamente, a meno che tutti i votanti abbiano votato;
- verificabilità universale e assenza di evidenza contemporaneamente, se la trascrizione del voto di un votante dipende dalla sua identità, il suo voto e la sola presenza di dati pubblici.

## 2.5 Applicazione della blockchain su sistemi di e-voting

Un importante caratteristica della tecnologia *blockchain* è che permetterebbe di fornire ai sistemi di votazione un database distribuito, data la sua natura decentralizzata, e quindi consentire una maggiore partecipazione del votante e una minore partecipazione di entità terze.

La tecnologia blockchain ha anche permesso che le informazioni riguardo ai voti possano venire registrate e rese pubbliche per tutti gli attori partecipanti a questo processo e garantendo inoltre che le informazioni una volta registrate non possano venire modificate.

Un importante aspetto, introdotto con la *Blockchain 2.0*<sup>1</sup>, è quello degli *smart contract*, dei contratti che permettono di specificare delle regole che possono essere usate per fornire una maggiore sicurezza nell'e-voting [3].

## 2.6 Applicazioni basate su blockchain presenti in letteratura

Di seguito (vedi Tabella 2.1) sono stati descritti dei sistemi di e-voting che sono implementati tramite la tecnologia blockchain, analizzandone le caratteristiche principali e facendo presenti eventuali problematiche e nel caso miglioramenti che vi si possono applicare.

Articolo	Caratteristiche	Note
[7]	Sistema di votazioni dove a ciascun candidato viene assegnata una blockchain diversa.	Grande complessità dovuta all'uso di una blockchain diversa per candidato.

(Continua)

---

<sup>1</sup><https://www.arvato-systems.com/blog/blockchain-smart-contracts>

Articolo	Caratteristiche	Note
[42]	Implementazione con scalabilità nazionale che utilizza due blockchain, una che conterrà le informazioni dell'utente votante e una per il voto dell'utente.	Possibile miglioramento delle prestazioni parallelizzando l'esecuzione delle transazioni.
[27]	Voto tramite l'uso delle Blind Signatures permettendo quindi al firmatario di firmare il voto senza che questo venga rivelato.	Grande latenza e ritardo all'interno di una implementazione su larga scala.
[47]	Utilizzo del framework blockchain Hyperledger Fabric, l'algoritmo di consenso Byzantine Fault Tolerance e le linkable ring signature per la firma dei voti.	Possibile miglioramento delle prestazioni parallelizzando l'esecuzione delle transazioni.
[23]	Utilizzo della blockchain Multichain, che limita ciascun utente votante ad una sola transazione. La validità del votante sarà verificata da un Trusted Third Party (TTP) attraverso un messaggio segreto inviatogli dal votante.	Grande latenza e ritardo causato dalla verifica del messaggio segreto che l'TTP deve compiere.
[24]	Utilizzo di una blockchain permissioned e setup di framework come Exonium, Quorum e Geth.	Implementazione costosa su larga scala se viene usato Exonium. Quorum e Geth non supportano l'esecuzione delle transazioni in parallelo.
[31]	Sistema in cui le informazioni dell'utente votante sono criptate tramite Secure Hash Algorithm (SHA). Il blocco del voto verrà aggiunto alla blockchain del candidato.	Grande complessità dovuta all'uso di una blockchain diversa per candidato.

(Continua)

Articolo	Caratteristiche	Note
[46]	Il sistema proposto utilizzerà la crittografia ellittica per criptare il voto dell'utente.	Utilizzo di un database PKI, il quale se violato può invalidare l'intero sistema.
[2]	Implementazione tramite tre componenti: rete blockchain, commissione elettorale e lato cliente.	Centralizzazione del sistema a favore della commissione elettorale.
[13]	Sistema realizzato tramite una struttura a livelli, dove ogni livello è connesso al suo superiore per garantire privacy e sicurezza.	Necessità di migliorare la sincronizzazione e la performance.
[20]	Sistema di votazioni implementato sulla blockchain Ethereum con l'uso di smart contract e crittografia omomorfica per la privacy.	Possibile miglioramento della verificabilità tramite l'algoritmo crittografico di Paillier.
[41]	Framework del sistema formato da: partecipanti, organizzatori, ispettori, algoritmo di crittografia, algoritmo di Hash(SHA-256), server di voto.	Grande latenza e ritardo all'interno di una implementazione su larga scala.
[22]	Proposta di sistema implementato tramite l'uso dell'algoritmo Shamir's Secret Sharing e di due blockchain, una per la memorizzazione dei voti e una per il conteggio.	Efficienza della fase di validazione e assegnazione del voto del candidato.
[34]	Sistema realizzato su rete peer-to-peer formato da: super-node, trusted nodes e seggi elettorali.	Possibile miglioramento delle prestazioni tramite implementazione su blockchain Ethereum con l'uso di smart contract.

Tabella 2.1: Analisi di vari sistemi di e-voting implementati su blockchain

# Capitolo 3

## Blockchain Ethereum

### 3.1 Blockchain

La blockchain può essere definita come una struttura dati *decentralizzata*, *distribuita* e *immutabile* costituita da dei blocchi contenenti le informazioni riguardanti le transazioni.

L'intero registro di transazioni sarà pubblico, permettendo quindi a chiunque di controllare le transazioni passate e quelle in corso. La blockchain dunque crescerà ogni volta che viene aggiunto un nuovo *blocco*, e ciascun blocco conterrà una *marca temporale* e un puntatore *hash* al blocco precedente, si formerà quindi una “*catena*” di blocchi (come riportato in Figura 3.1).

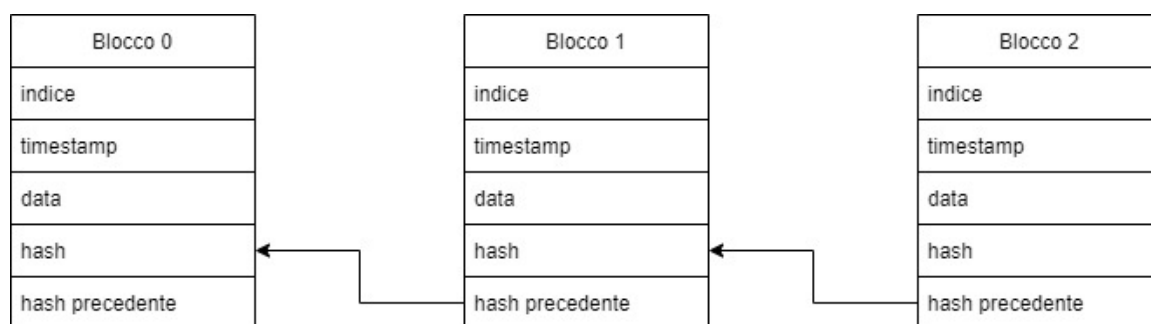


Figura 3.1: Struttura dei blocchi all'interno di una blockchain

Non sarà quindi possibile inserire modificare o eliminare un blocco senza che ciò venga rivelato, infatti un cambiamento in qualsiasi blocco invaliderebbe tutti i

blocchi successivi, dato che gli hash dei blocchi successivi cambierebbero a loro volta.

I blocchi saranno identici all'interno della *rete peer-to-peer* e per aggiungere un nuovo blocco tutti i nodi della rete dovranno raggiungere un accordo confermando la legittimità di questa operazione utilizzando un *algoritmo di consenso*, senza necessita quindi di un intermediario che esegua e convalidi le transazioni.

Le principali caratteristiche dell'architettura blockchain sono quindi:

- *Crittografia*: le transazioni sono legate e verificate tramite algoritmi crittografici.
- *Immutabilità*: i blocchi non potranno essere modificati o eliminati.
- *Provenienza*: possibilità di tracciare ogni transazione.
- *Decentralizzazione*: il registro di transazioni sarà accessibile da tutti i nodi partecipanti, e le modifiche saranno possibili tramite l'uso di un algoritmo di consenso.
- *Anonimità*: l'indirizzo pubblico non farà riferimento all'identità di un utente.

## 3.2 Ethereum

Ethereum è una infrastruttura decentralizzata, open source, che fa uso della tecnologia blockchain per memorizzare e sincronizzare i cambiamenti di stato all'interno del sistema [5].

La piattaforma Ethereum è stata creata da Vitalik Buterin [14], per cercare di risolvere delle limitazioni presenti nel protocollo Bitcoin: tipi di transazione, tipi di dato e grandezza di data.

Le blockchain di Ethereum e Bitcoin condividono diverse caratteristiche come può essere l'uso di una rete peer-to-peer, un algoritmo di consenso Proof of Work, primitive crittografiche e una moneta digitale.

Ethereum, a differenza di Bitcoin, è una blockchain che permette l'esecuzione di programmi complessi, smart contract, tramite una macchina virtuale, *Ethereum Virtual Machine* (EVM). Per di più permette di tracciare, oltre ai movimenti di criptomonete, lo stato delle transazioni e di basi di dati.

La principale caratteristica, che lo ha reso popolare e seconda solo a Bitcoin in termini di notorietà e capitale, è la possibilità per gli sviluppatori di creare delle App

Decentralizzate, *dApp*, tramite degli *smart contract* scritti utilizzando un linguaggio *turing-equivalent*, e quindi permettendo il supporto di tutti i tipi di operazioni. Le componenti della blockchain Ethereum sono:

- Una rete P2P, *Ethereum main network*, che usa la porta 30303 ed esegue un protocollo detto *DEVP2p*.
- Un algoritmo di consenso detto Proof of Work che permette di raggiungere un accordo, tra tutti i nodi della rete, sullo stato delle informazioni memorizzate nella blockchain di Ethereum.
- Le transazioni, messaggi nella rete che comprendono informazioni come mittente, destinatario, dati...
- La Ethereum Virtual Machine che esegue gli smart contract.
- Una struttura dati, composta da delle strutture dati, chiamate Merkle Patricia Tree, che memorizzeranno ciascuno lo stato della rete.

### 3.3 Smart contract

Gli smart contract sono dei programmi, installati nella blockchain Ethereum, costituiti da dei set di regole, scritti in codice, che per funzionare devono essere eseguiti dalla Ethereum Virtual Machine.

Questi contratti verranno associati a degli account Ethereum, che non possiedono chiavi private, quindi non sono controllati da nessuna entità, se non dal codice del programma, più precisamente da delle funzioni che verranno eseguite nel caso determinate condizioni prestabilite vengano soddisfatte [39].

Ogni smart contract verrà identificato da un indirizzo Ethereum, che potrà essere utilizzato come destinatario in una transazione, per inviare fondi al contratto o chiamare delle funzioni nel contratto.

L'esecuzione degli smart contract, quindi dipenderà da una chiamata esterna di una transazione, i contratti non potranno eseguire funzioni per conto proprio o effettuare delle esecuzioni in background. Le transazioni sono atomiche, perciò verranno terminate con successo oppure no. Il loro utilizzo garantirà quindi l'esecuzione di operazioni in sicurezza, velocità, autonomia e trasparenza.

La Ethereum Virtual Machine è una macchina virtuale in grado di comprendere del codice scritto in un linguaggio chiamato EVM bytecode. È possibile scrivere il codice degli smart contract direttamente in EVM bytecode, essendo però un linguaggio complesso da leggere e capire, i programmatori preferiscono l'utilizzo di un linguaggio ad alto livello per la scrittura del codice del programma, e poi convertirlo in bytecode tramite un compilatore.

Benché sia possibile convertire qualunque linguaggio di alto livello in bytecode, è una operazione complessa, è più semplice invece fare uso di un linguaggio ad alto livello creato appositamente per la scrittura di smart contract.

### 3.4 Solidity

Solidity è il linguaggio di programmazione più popolare e più frequentemente utilizzato dai programmatori nella creazione di smart contract su Ethereum e altre blockchain [30].

Solidity è un linguaggio di tipo statico e procedurale, è stato disegnato facendo riferimento ad una sintassi ECMAScript, però a differenza di questo, accetterà valori di ritorno di tipo statico o variabile. La sintassi avrà una struttura con elementi simili a JavaScript, C++ e Java.

Solidity fu inizialmente progettato da Gavin Wood, con l'obiettivo di creare un linguaggio di programmazione per lo sviluppo di smart contract da poter eseguire sulla blockchain Ethereum dalla Ethereum Virtual Machine. Successivamente, grazie alle sue caratteristiche, questo linguaggio è stato riutilizzato per la scrittura di smart contract tra diverse Blockchain [5]. Attualmente Solidity è un progetto sviluppato in maniera indipendente su GitHub<sup>1</sup>.

I contratti supporteranno l'ereditarietà, includendo l'ereditarietà multipla tramite la Linearizzazione C3, un algoritmo utilizzato per ordinare correttamente i metodi in caso di ereditarietà multipla.

L'applicazione principale del progetto Solidity è il compilatore Solidity (solc) che effettuerà la conversione degli smart contract scritti in Solidity in EVM bytecode.

---

<sup>1</sup><https://github.com/ethereum/solidity>



Inoltre farà uso di una *application binary interface* (ABI) per garantire la type safety (sicurezza rispetto ai tipi), nel caso quindi il compilatore rilevasse una incompatibilità tra tipo di dato per ogni variabile, allora l'ABI genererà un errore.

## 3.5 dApp

Un'*applicazione decentralizzata* (dApp) è un'applicazione che collega uno smart contract con un'interfaccia utente frontend [33]. Una dApp, per essere definita tale, dovrà essere:

- *Decentralizzata*: funzionamento all'interno della blockchain, dove nessuna entità detiene il controllo.
- *Deterministica*: ad un certo input corrisponderà sempre lo stesso output.
- *Turing-complete*: i comandi potranno venire eseguiti una volta soddisfatti i requisiti necessari.
- *Isolata*: esecuzione su un ambiente virtuale detto *Ethereum Virtual Machine* (EVM), in questa maniera in caso di malfunzionamento la rete blockchain non verrà intaccata.

## 3.6 Nodi Ethereum

Il registro delle transazioni della blockchain Ethereum sarà accessibile, tramite un software client, a tutti i nodi partecipanti alla rete Ethereum. I nodi presenti nella rete Ethereum possono essere delle seguenti tipologie:

- *Nodi completi*: memorizzano e verificano tutte le transazioni sulla blockchain Ethereum, oltre a poter interagire direttamente con gli smart contract. Affinché siano operativi è però richiesta una sincronizzazione e manutenzione costante del nodo con quella che è la rete Ethereum, venendo quindi a gravare sulle risorse hardware del computer che esegue il nodo e sul consumo di larghezza di banda.

- *Nodi leggeri*: memorizzano solo in parte il registro della blockchain ed eventualmente riceveranno dati aggiuntivi su richiesta. Sarà quindi possibile eseguire i nodi su dei dispositivi meno potenti e non sarà necessaria la manutenzione richiesta dai nodi completi.
- *Nodi di archivio*: memorizzano l'intero registro di transazioni della blockchain Ethereum, costruendo un archivio di stati storici della blockchain.

## 3.7 Account Ethereum

Gli account Ethereum sono costituiti da degli indirizzi di 20 byte, e possono essere di due tipi: di proprietà esterna (controllato da chiunque possieda le chiave private) e contratto (controllato dal codice dello smart contract) [44]. Un account Ethereum sarà formato da quattro campi:

- *Nonce*: indica il numero di transazioni inviate oppure il numero di contratti creati da un account.
- *Balance*: numero di *wei* posseduti da un account, dove *wei* è la più piccola frazione di Ether, tale che  $1 \text{ ETH} = 10^{18} \text{ wei}$ .
- *CodeHash*: hash del codice di un account sulla EVM, che verrà eseguito nel caso un account riceva una chiamata di messaggio.
- *StorageRoot*: hash a 256 bit del nodo radice di un *albero di Merkle Patricia* che codifica il contenuto dello spazio di archiviazione dell'account.

## 3.8 Transazioni Ethereum

Per transazione si intende un'istruzione, che cambi lo stato della rete, iniziata da un *externally owned account* [19].

Una transazione conterrà i seguenti dati:

- *Recipient*: indirizzo Ethereum del destinatario della transazione.

- *Signature*: firma generata dalla chiave privata del mittente nel momento in cui concede l'autorizzazione per la transazione.
- *Value*: importo in wei da trasferire dal mittente al destinatario.
- *Data*: dati facoltativi che si possono aggiungere.
- *Gaslimit*: importo massimo di unità di gas che potranno essere consumate durante la transazione.
- *MaxPriorityFeePerGas*: importo massima di ricompensa che si include come ricompensa al miner.
- *MaxFeePerGas*: importo massimo di gas che si è disposti a pagare per il mining della transazione.

### 3.9 Algoritmi di consenso: Proof of Work

Un algoritmo di consenso è una procedura tramite la quale i peer all'interno di una rete blockchain raggiungano un accordo riguardo allo stato attuale della blockchain. In questo modo verrà ottenuta la fiducia tra i peer, nella rete decentralizzata, senza bisogno di un'autorità. Il protocollo di consenso consentirà quindi di aggiungere un nuovo blocco nella blockchain nel caso ci sia il consenso da parte di tutti i nodi, l'algoritmo di consenso usato attualmente da Ethereum è il *Proof of Work* (PoW) [25].

L'idea su cui si basa questo algoritmo è che dei nodi detti *miner*, facendo uso di hardware specializzato, competeranno per essere i prossimi ad aggiungere un blocco nella blockchain, per fare ciò dovranno cercare di risolvere un complesso problema matematico, questo richiederà un elevato consumo di potenza computazionale (e quindi energia elettrica), ma il primo miner che riuscirà a risolvere il problema verrà premiato con la *fee* della transazione relativa a quel blocco e una ricompensa in ether detta "*block reward*". In fine il miner procederà a condividere il nuovo blocco con il resto della rete, che lo accetteranno come blocco successivo canonico della blockchain.

### 3.10 Ether

Ether (ETH) è la criptovaluta nativa di Ethereum. Il suo scopo primario è quello di incentivare economicamente affinché i nodi della rete verifichino le transazioni [32].

Infatti ogni partecipante che intende trasmettere una richiesta di transazione dovrà fornire un determinato importo di ether come ricompensa per il miner che riuscirà a verificare la transazione e trasmetterla nella rete.

L'importo sarà proporzionale al tempo impiegato per l'esecuzione della transazione, questo servirà come disincentivo per gli eventuali utenti malintenzionati che cercassero di intasare la rete tramite l'esecuzione di operazioni computazionalmente complesse, dovrebbero infatti pagare per il tempo di calcolo impiegato.

### 3.11 Token ERC20

ERC-20 (*Ethereum Request for Comments 20*) è uno standard token, proposto da Fabian Vogelsteller [43], che implementa un'API per token fungibili (intercambiabili tra di loro) all'interno di smart contract.

Questo standard fornisce funzionalità basiche per trasferire token e permetterne l'utilizzo tramite altre applicazioni Ethereum, tramite sei funzioni e due eventi che devono essere implementati in un contratto:

- `totalSupply()` è una funzione che restituisce come output la quantità di token esistenti.
- `balanceOf(account)` è una funzione che restituisce come output la quantità di token posseduti da un indirizzo passato in input (*account*).
- `allowance(owner, spender)` è una funzione che restituisce come output il numero di token che un indirizzo (*spender*) può spendere per conto di un altro indirizzo (*owner*).
- `transfer(recipient, amount)` è una funzione che restituisce un booleano che indicherà il successo o meno di un trasferimento di N (*amount*) token ad un indirizzo (*recipient*).

- `approve(spender, amount)` è una funzione che restituisce un booleano che indicherà il successo o meno dell'operazione di permesso da parte di un indirizzo (*spender*) di spendere N (*amount*) token.
- `transferFrom(sender, recipient, amount)` è una funzione che restituisce un booleano che indicherà il successo o meno dell'operazione di trasferimento di N (*amount*) token da un indirizzo (*sender*) ad un indirizzo (*recipient*).
- `Transfer(from, to, value)` è un evento emesso quando una quantità (*value*) di token viene trasferita da un indirizzo (*from*) ad un indirizzo (*to*).
- `Approval(owner, spender, value)` è un evento emesso quando avverrà con successo la chiamata ad una funzione *approve*.

Se si vuole creare un token ERC20 si dovrà quindi programmarlo implementando una interfaccia (come riportato in Listato 3.1) che faccia uso delle funzioni ed eventi sopra descritti.

```

contract ERC20Interface {
    function totalSupply() public constant returns (uint);
    function balanceOf(address tokenOwner) public constant
        returns (uint balance);
    function allowance(address tokenOwner, address spender)
        public constant returns (uint remaining);
    function transfer(address to, uint tokens) public returns
        (bool success);
    function approve(address spender, uint tokens) public
        returns (bool success);
    function transferFrom(address from, address to, uint
        tokens) public returns (bool success);

    event Transfer(address indexed from, address indexed to,
        uint tokens);
    event Approval(address indexed tokenOwner, address
        indexed spender, uint tokens);
}

```

Listato 3.1: Interfaccia token ERC20

## 3.12 Network

Il network principale di Ethereum viene chiamato *mainnet*, quello completamente testato e supportato dagli sviluppatori e dalla comunità, dove ether e i vari token hanno un valore monetario reale [40]. Un Ethereum *testnet* è una rete utilizzata per effettuare dei test, di smart contract e protocolli, all'interno di un ambiente controllato. L'installazione degli smart contract, così come anche interagire con questi tramite le chiamate di funzioni, sono operazioni che richiedono un pagamento di fee in ether.

Per evitare di spendere ether e quindi reale denaro, si fa utilizzo dei testnet, dove si potrà ottenere ether e altre criptomonete in modo gratuito. La maggior parte dei testnet usa un algoritmi di consenso di tipo Proof of Authority (PoA) [36], dove solo un ristretto numero di nodi, detti *validatori* si occuperà di validare le transazioni e creare nuovi blocchi.

# Capitolo 4

## Tornado Cash

Tornado Cash<sup>1</sup> è un protocollo *open-source*, *decentralizzato* e *non-custodial* implementato sulla blockchain Ethereum (come descritto in Sezione 4.1).

La privacy solution implementata da Tornado Cash [35], utilizza degli *smart contract* che permettono il deposito di ether o altri ERC20 tokens da parte di un indirizzo e il ritiro da un indirizzo diverso, in modo completamente anonimo (come riportato in Sezione 4.4).

I depositi e i ritiri sono possibili tra valori prefissati per garantire maggiore anonimità; attualmente Tornado Cash supporta 10 token diversi (vedere Sezione 4.2), ad esempio ETH dispone di quattro pool per l'importo di: 0.1, 1, 10 e 100 ETH.

### 4.1 Protocollo decentralizzato e non-custodial

Gli smart contract di Tornado Cash essendo implementati sulla *blockchain* di Ethereum non possono essere né modificati né manomessi. Gli smart contract di mining e quelli di amministrazione sono implementati dalla comunità in maniera decentralizzata: un qualsiasi utente potrà fare una proposta che consisterà in uno smart contract e alla quale si potrà votare in modo favorevole o contrario tramite il locking di TORN (token di Tornado Cash), dopo 5 giorni se è stato raggiunto un minimo di 25000 TORN e la proposta è stata votata per la maggioranza di voti a favore, allora questa potrà essere eseguita e modificare quelli che sono gli smart contract di mining e amministrazione.

---

<sup>1</sup><https://tornado.cash/>

Gli utenti avranno la completa *custodia e controllo* dei loro token mentre operano su Tornado Cash, questo sarà permesso loro tramite una nota privata fornita loro al momento del deposito [8].

## 4.2 Blockchain operative

Attualmente Tornado Cash è operativo all'interno di [8]:

- *Ethereum Blockchain*: ETH (Ethereum), DAI (Dai), cDAI (Compound Dai), USDC (USD Coin), USDT (Tether) & WBTC (Wrapped Bitcoin);
- *Binance Smart Chain*: BNB (Binance Coin);
- *Polygon Network*: MATIC (Polygon);
- *Gnosis Chain*: xDAI (xDai);
- *Avalanche Mainnet*: AVAX (Avalanche);
- *Optimism*, come Layer-2 per ETH;
- *Arbitrum One*, come Layer-2 per ETH.

## 4.3 Setup

Di seguito verrà descritto il setup implementato per il protocollo di Tornado Cash [35]. Sia  $\mathbb{B} = \{0,1\}$ . Sia  $e$  l'operazione di abbinamento usata in SNARK, una implementazione crittografica che permette la dimostrazione di una dichiarazione (vedere Sezione 4.5), che è definita su gruppi di numeri primi  $q$ . Sia  $H_1 : \mathbb{B}^* \rightarrow \mathbb{Z}_p$  la funzione di hash Pedersen. Sia  $H_2 : (\mathbb{Z}_p, \mathbb{Z}_p) \rightarrow \mathbb{Z}_p$  la funzione hash MiMC, una funzione hash disegnata specificamente per le applicazioni SNARK [4]. Sia  $\mathcal{T}$  un albero di Merkle di altezza 20, dove ogni nodo non foglia fa l'hash dei suoi due figli con  $H_2$ ; inizialmente le foglie avranno valore 0, che successivamente saranno sostituiti con dei valori di  $\mathbb{Z}_p$ . Sia  $O(\mathcal{T}, l)$  l'inizio della foglia con indice  $l$  dell'albero  $\mathcal{T}$ . Sia  $k \in \mathbb{B}^{248}$  un nullifier e  $r \in \mathbb{B}^{248}$  un segreto. Sia l'indirizzo Ethereum del destinatario del deposito



A. Sia  $\mathcal{S}[R, h, A, f, t]$  tale che  $[R, h, A, f, t]$  siano dei valori pubblici dove  $h$  è l'hash del nullifier e  $\parallel$  la concatenazione di sequenze di bit:

$\mathcal{S}[R, h, A, f, t] = \{k, r \in \mathbb{B}^{248}, l \in \mathbb{B}^{16}, O \in \mathbb{Z}_p^{16} \text{ tale che } h = H_1(k) \ \& \ O \text{ sia l'inizio di } H_2(k \parallel r) \text{ in posizione } l \text{ di } R\}$

Sia  $\mathcal{D} = (d_p, d_v)$  la chiave di prova-verifica di ZK-SNARK per  $\mathcal{S}$ .

Sia  $\text{Prove}(d_p, \mathcal{T}, k, r, l, A, f, t) \rightarrow P$  la prova e  $\text{Verify}(d_v, P, R, h, A, f, t)$  il verificatore della prova. Sia  $\mathcal{C}$  lo smart contract con le seguenti funzionalità:

- Memorizzerà gli ultimi  $n = 100$  valori della radice e i valori dei nodi nel percorso dall'ultima foglia aggiunta fino alla radice necessari per il calcolo della prossima radice dell'albero di Merkle  $\mathcal{T}$ .
- Accetta pagamento di  $N$  token con parametro  $C \in \mathbb{Z}_p$ . Il valore  $C$  sarà aggiunto all'albero di Merkle, la radice sarà ricalcolata e la radice precedente memorizzata in un array.
- Verifica la prova  $P$  rispetto ai valori pubblici  $(R, h, A, f, t)$ . Se la verifica ha successo il contratto invierà all'indirizzo  $A$   $(N - f)$  token e  $f$  token al relayer con indirizzo  $t$ .
- Viene verificato che il ritiro sia valido controllando che l'hash del nullifier della prova non sia già presente all'interno della lista degli hash dei nullifier, e quindi verrà poi aggiunto in questa ultima.

## 4.4 Depositi e ritiri

La procedura che un utente dovrà seguire per poter effettuare un deposito (come illustrato in Figura 4.1) consisterà innanzitutto nell'effettuare il login, tramite wallet, sull'applicazione web di Tornado Cash<sup>2</sup>, si potrà poi selezionare tra le opzioni presenti il token e la quantità che si intende depositare.

Se si ha disponibilità nel proprio wallet, del token nella quantità selezionata, allora verrà eseguita la funzione di deposito sullo smart contract, e si riceverà una nota privata che si potrà poi utilizzare per il ritiro dei fondi depositati.

---

<sup>2</sup><https://tornadocash.eth.limo/>

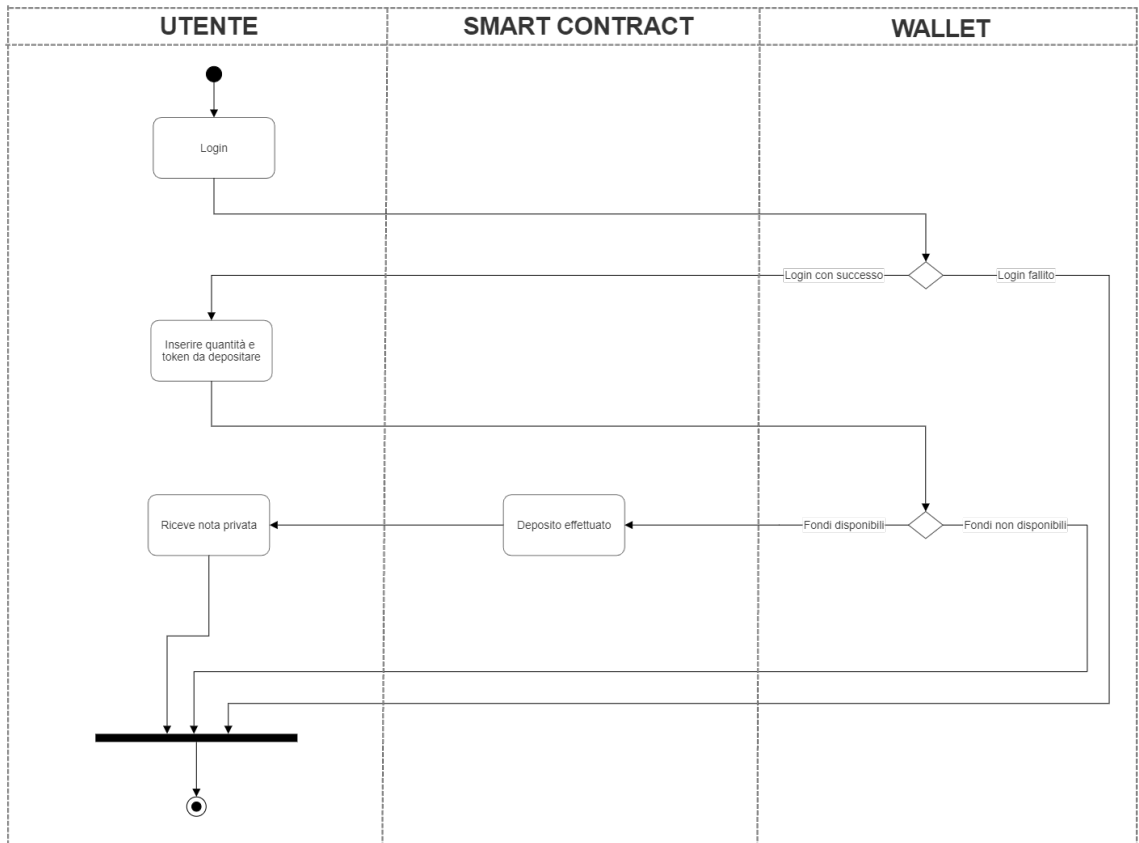


Figura 4.1: Diagramma di attività: Deposito

L'utente che dopo aver fatto un deposito vorrà effettuare il corrispettivo ritiro (come rappresentato in Figura 4.2), collegandosi sempre alla pagina web di Tornado Cash<sup>3</sup>, dovrà inserire la nota privata che ha ottenuto al momento del deposito e l'indirizzo Ethereum a cui si vuole venga destinato il ritiro.

Dovrà poi scegliere se procedere con il ritiro effettuando il login tramite wallet e quindi utilizzare l'eventuale ether presente all'interno di esso per il pagamento della fee necessaria per la transazione oppure se usufruire del servizio di un relay e quindi si occupi lui di pagare la fee richiesta nel ritiro in cambio di una piccola percentuale presa dall'importo che si vuole ritirare. L'utente potrà quindi scegliere in che modo effettuare il ritiro:

- Collegando un proprio *wallet* (Metamask o WalletConnect) al sito web di Tornado Cash e utilizzando questo per pagare la fee necessaria per il ritiro dell'importo

<sup>3</sup><https://tornadocash.eth.limo/>

depositato.

Si effettuerà una transazione al contratto  $\mathcal{C}$  passando come input  $R, h, A, f, t, P$  tale che  $R$  indichi la radice del Merkle tree,  $h$  l'hash del nullifier,  $A$  l'indirizzo del destinatario,  $f$  la fee della transazione,  $t$  l'indirizzo del relayer e  $P$  la prova del protocollo zk-SNARK.

Si dovrà quindi avere su questo wallet degli ether, e per garantire l'anonimità si consiglia che questi vi siano stati trasferiti tramite l'uso di un relayer.

- Utilizzando un *relayer*, che permetterà di effettuare il ritiro su un qualsiasi indirizzo Ethereum, anche uno nuovo, senza necessita di effettuare il collegamento del wallet sul sito web di Tornado Cash, in quanto sarà il relayer ad occuparsi di pagare la fee per l'operazione, e in cambio riceverà una piccola parte del deposito.

Si invierà quindi al relayer una richiesta di transazione al contratto  $\mathcal{C}$  passando come input  $R, h, A, f, t, P$ .

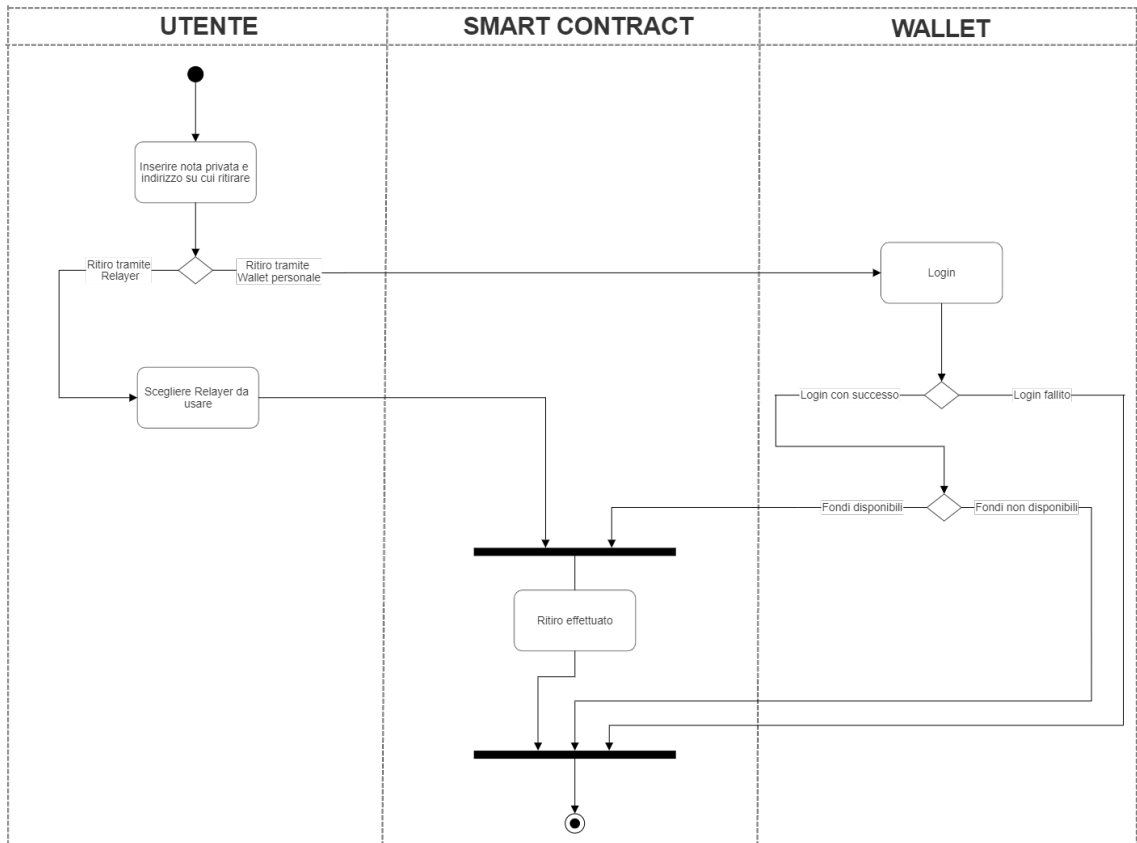


Figura 4.2: Diagramma di attività: Ritiro

Le azioni di deposito e ritiro da parte dell'utente avvengono interagendo con lo smart contract del Proxy<sup>4</sup> di Tornado Cash.

In Figura 4.3 sono state rappresentate la procedura di deposito e ritiro, indicando le funzioni che ne determinano l'esecuzione e le interazioni tra le varie entità del sistema. Di seguito verranno descritte in maniera dettagliata in che modo operano le funzioni coinvolte.

L'utente che vuole fare un deposito<sup>5</sup> all'interno del *pool*<sup>6</sup>, chiamerà quindi la funzione:

<sup>4</sup>Indirizzo proxy di Tornado Cash:

<https://goerli.etherscan.io/address/0x454d870a72e29d5e5697f635128d18077bd04c60>

<sup>5</sup>Deposito di 1 ETH su pool di Tornado Cash:

<https://goerli.etherscan.io/tx/0x930c7407516ddf06eb6c553c19ffff1588e03c2b93eaf045b4a8da472bfd981a>

<sup>6</sup>Indirizzo pool di 1 ETH di Tornado Cash:

<https://goerli.etherscan.io/address/0x3aac1cc67c2ec5db4ea850957b967ba153ad6279>

```
deposit(address _tornado, bytes32 _commitment, bytes
        _encryptedNote)
```

L'utente dovrà allora generare in modo randomico un *segreto* e un *nullifier* rispettivamente  $k, r \in \mathbb{B}^{248}$ , e il suo *hash*, detto *commitment*  $C$  (parametro `_commitment`), tale che  $C = H_1(k||r)$ , verrà inviato insieme ad  $N$  token allo smart contract  $\mathcal{C}$  (parametro `_tornado`). Il contratto quindi accetterà il deposito e aggiungerà  $C$  come una foglia dell'albero, nel caso questo non sia pieno [15].

La funzione per effettuare un ritiro<sup>7</sup> sarà:

```
withdraw(address _tornado, bytes _proof, bytes32 _root,
        bytes32 _nullifierHash, address _recipient, address
        _relayer, uint256 _fee, uint256 _refund)
```

Per effettuare il ritiro l'utente dovrà selezionare l'indirizzo  $A$  del destinatario (parametro `_recipient`) con una commissione di transazione (parametro `_fee`) tale che  $f \leq N$ , dovrà poi dimostrare, tramite una prova, di possedere un segreto al quale corrisponde un commitment, memorizzato nella lista dello smart contract, il quale però non sia già stato utilizzato per fare un ritiro. Questa prova però non dovrà rivelare il deposito corrispondente al segreto, viene quindi utilizzato un protocollo, in crittografia detto *zero-knowledge proof* (come vedremo in Sezione 4.5).

Verrà poi selezionata una radice  $R$  (parametro `_root`) tra quelle memorizzate sul contratto e calcolata l'apertura  $O(l)$  che termina con  $R$ . Si calcolerà l'hash del nullifier  $h = H_1(k)$  (parametro `_nullifierHash`) e la prova  $P$  (parametro `_proof`) chiamando Prove su  $d_p$ .

Lo smart contract, una volta verificata la prova e l'unicità dell'hash del nullifier, procederà a trasferire  $(N - f)$  token ad  $A$  e  $f$  token al relayer  $t$  (parametro `_relayer`), aggiungerà poi  $h$  alla lista degli hash dei nullifier.

---

<sup>7</sup>Ritiro di 1 ETH tramite relayer dal pool di Tornado Cash:  
<https://goerli.etherscan.io/tx/0xf59671213cc17c5c7fc01ce3550626ab88431c285682f8cca03f760f20adb0be>  
Ritiro di 0.1 ETH tramite Wallet personale dal pool di Tornado Cash:  
<https://goerli.etherscan.io/tx/0x293fe96f9b17738437aae13e63be0e706b91b489238008589f297e1144ce9eb8>

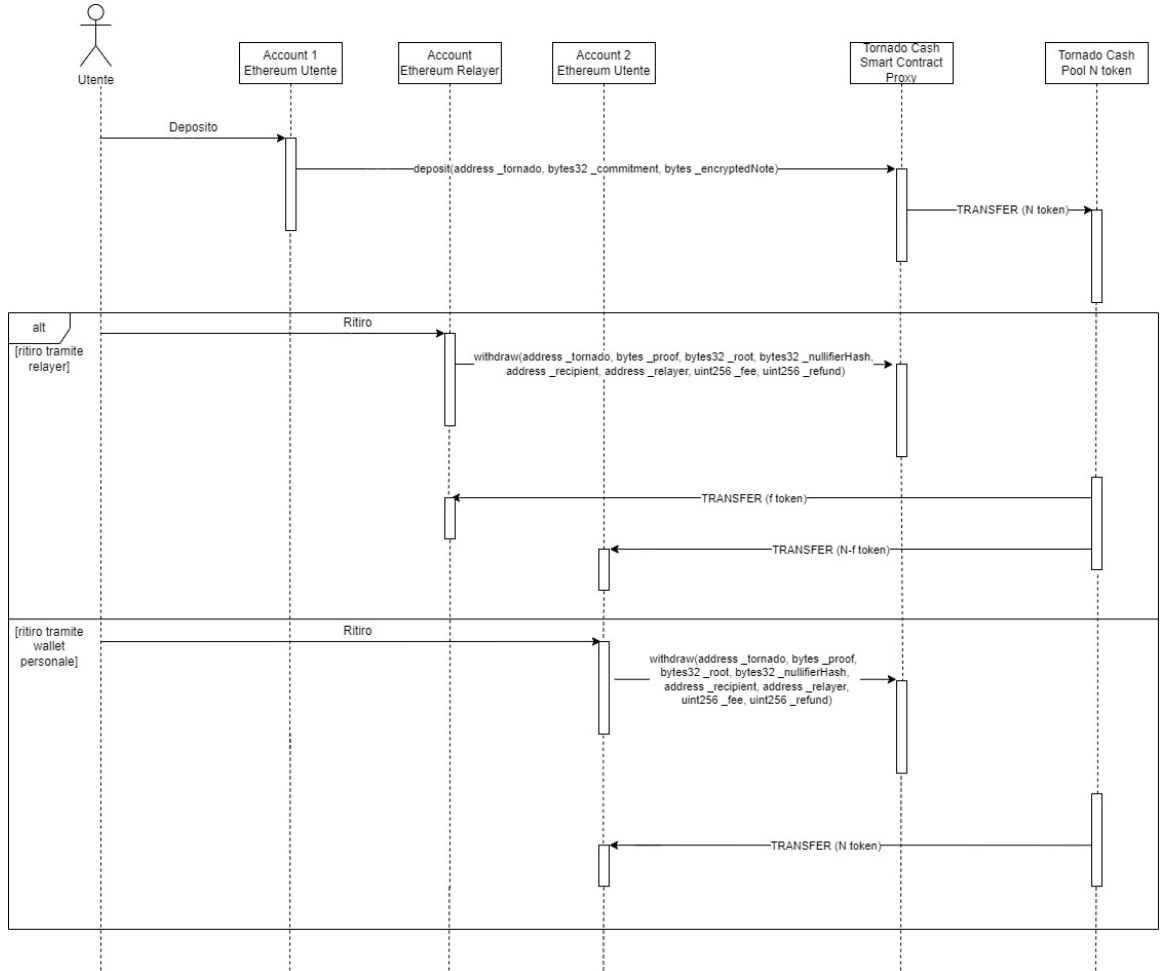


Figura 4.3: Diagramma di sequenza: Deposito e Ritiro

## 4.5 zk-SNARK proof

L'anonimità è resa possibile tramite *zk-SNARK* che sta per “*Zero-Knowledge Succinct Non-Interactive Argument of Knowledge*”, una implementazione crittografica che garantisce la zero-knowledge proof, una tecnica che permette ad una entità (*prover*) di dimostrare ad un'altra (*verifier*) che una dichiarazione è vera senza rivelare nessun tipo di informazione oltre all'autenticità della dichiarazione stessa [1].

Quando viene fatto un deposito viene generata un'area randomica di byte, di questa verrà poi eseguito il *Pedersen Hash*, un algoritmo di hashing spesso utilizzato all'interno di protocolli zero-knowledge proof (come descritto in Sezione 4.6).

Quindi l'hash generato verrà inviato insieme al token allo smart contract, il quale memorizzerà l'hash all'interno di un *Merkle tree* (vedere Sezione 4.7).

Quando viene fatto un ritiro, questa area di byte sarà formata da una parte detta *segreto* e una detta *nullifier*. Questi saranno dei numeri interi randomici, ciascuno lungo 31 bytes.

Il nullifier verrà cifrato e inviato allo smart contract, che controllerà nello smart contract la validità dell'hash fornito [9].

## 4.6 Pedersen Hash

La funzione di hash Pedersen associa una sequenza di bit ad un punto compresso in una curva ellittica [10]. L'efficienza nel calcolo della funzione di hashing che garantisce l'implementazione di Pedersen risulta particolarmente adatto ai circuiti zk-SNARK. Sia  $M$  una sequenza di bit, allora la funzione di hashing Pedersen di  $M$  è definita nel seguente modo:

- Siano  $P_0, P_1, \dots, P_k$  dei generatori uniformemente generati di  $\mathbb{G}$  (per un intero  $k$ );
- Dividere  $M$  in sequenze di al più 200 bits e ciascuna in parti di al più 4 bits.

$$M = M_0 M_1 \dots M_l \quad \text{dove} \quad M_i = m_0 m_1 \dots m_{k_i} \quad \text{con} \quad \begin{cases} k_i = 49 & \text{per } i = 0, \dots, l-1, \\ k_i \leq 49 & \text{per } i = l, \end{cases} \quad (4.1)$$

dove i termini  $m_j$  sono formati da parti di 4 bits

$$enc(m_j) = (2b_3 - 1) \cdot (1 + b_0 + 2b_1 + 4b_2) \quad (4.2)$$

$$\langle M_i \rangle = \sum_{j=0}^{k_i-1} enc(m_j) \cdot 2^{5j}. \quad (4.3)$$

La funzione di hashing Pedersen per  $M$  sarà

$$H(M) = \langle M_0 \rangle \cdot P_0 + \langle M_1 \rangle \cdot P_1 + \langle M_2 \rangle \cdot P_2 + \dots + \langle M_l \rangle \cdot P_l. \quad (4.4)$$

## 4.7 Merkle Tree

Un albero di Merkle [28] è un albero in cui ogni nodo *foglia* viene etichettato con l'hash di un dato, mentre gli altri *nodì*, non foglie, vengono etichettati con l'hash dei suoi nodi figli. Il nodo *radice* conterrà l'hash di tutti i nodi.

Nell'implementazione di Tornado Cash, quando viene effettuato un deposito, si inviano dei token e un hash allo smart contract, questo hash verrà memorizzato all'interno delle foglie di un albero di Merkle. Al ritiro, sarà necessario dimostrare di possedere un hash che sia memorizzato all'interno dell'albero di Merkle, per fare ciò si dovrà fornire una lista di hash che combinata restituirà la radice del merkle tree, questa prova fornita al *zk-snark* costituirà la *zero knowledge proof*.

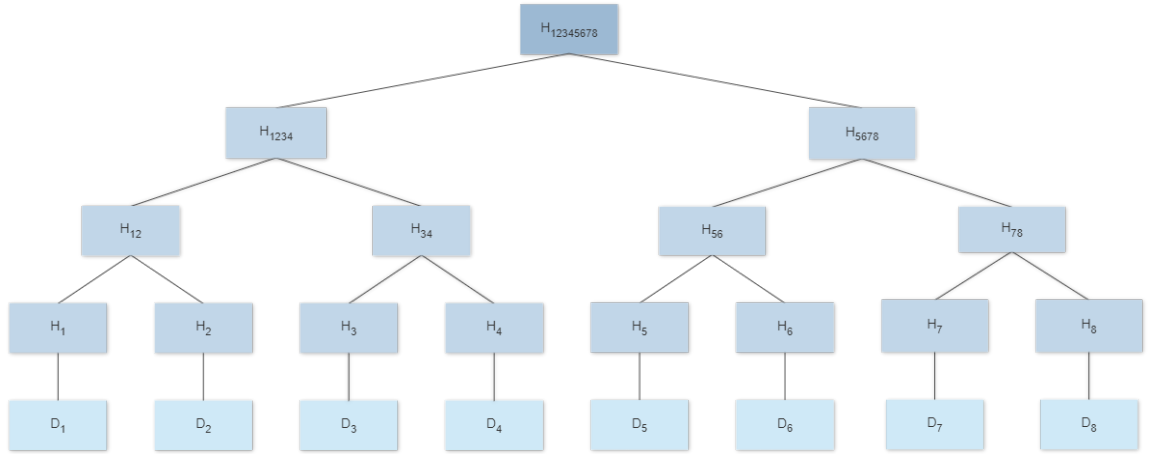


Figura 4.4: Esempio di albero di Merkle binario, dove  $D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8$  indicano dei dati, e  $H_1, H_2, H_3, H_4, H_5, H_6, H_7, H_8$  i corrispettivi valori hash

Data la radice dell'albero di Merkle  $H_{12345678}$  (come rappresentata in Figura 4.4) per dimostrare che uno dei valori tra  $H_1, H_2, H_3, H_4, H_5, H_6, H_7, H_8$  appartiene all'albero, sarà necessario fornire una *Merkle proof*, in modo tale da dimostrare che gli hash dell'albero non siano stati corrotti (vedi Figura 4.5).



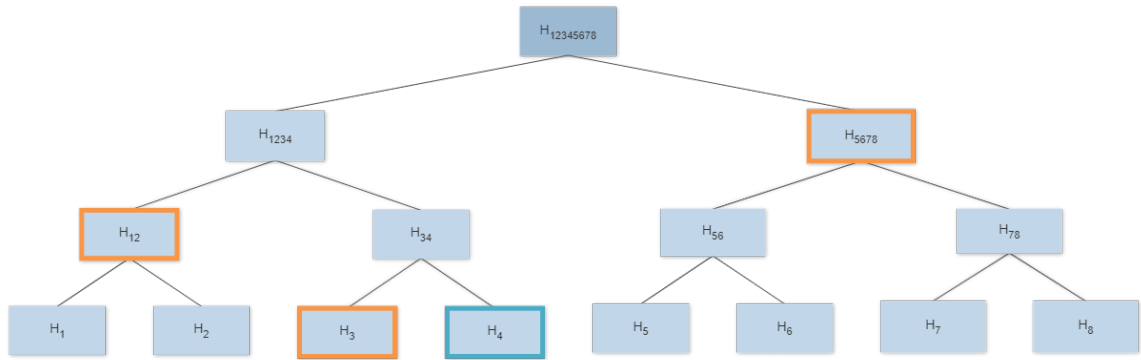


Figura 4.5: Per esempio, per dimostrare che  $H_4$  appartenga all'albero di Merkle la lista di hash che si dovrà fornire per ottenere la radice sarà  $H_3, H_{12}, H_{5678}$

## 4.8 Anonimità

Degli accorgimenti da tenere in considerazione per quanto riguarda la propria anonimità off-chain [12] sono:

- Memorizzare in un posto sicuro la propria nota privata, ed eliminarla una volta terminato il ritiro; questa può essere infatti utilizzata per ritirare i fondi depositati o per conoscere l'indirizzo usato per fare il deposito;
- Eliminare i dati dal proprio browser dopo ogni transazione in modo tale da evitare che questi possano essere ricondotti allo stesso utente, se possibile cambiare quindi browser, wallet e IP;
- Utilizzare dei servizi come vpn, proxies, Tor per nascondere il proprio IP da entità quali il proprio ISP, che potrebbero tracciare le attività correlate all'IP.

Delle precauzioni per mantenere la propria identità anonima in blockchain [16], ed evitare la correlazione tra il deposito e il ritiro, sono:

- Effettuare ritiri usando un relayer: sarà questo a pagare la fee per la transazione preservandone quindi la anonimità.

- Aspettare diverse transazioni da parte di altri utenti prima di effettuare il ritiro: se sono stati eseguiti un deposito e un ritiro di seguito è probabile che questi siano collegati tra loro.
- Lasciare trascorrere del tempo dal momento nel quale si effettua il deposito al momento nel quale si effettua il ritiro: anche nel caso siano presenti multipli depositi dopo il proprio, questi potrebbero essere stati emessi da un singolo utente per creare un falso senso di anonimità nel pool, è bene quindi aspettare anche più di un giorno per assicurarsi che siano effettuati depositi da molteplici utenti.
- Evitare di effettuare operazioni di deposito e di ritiro solo durante le ore di veglia del proprio fuso orario: questo ridurrebbe l'anonimità di un account rendendolo geolocalizzabile e rendendo possibili delle associazioni tra depositi e ritiri se queste operazioni vengono eseguite nei soliti stessi orari.
- Evitare l'uso degli stessi servizi tra l'indirizzo di deposito e quello di ritiro.

Altre precauzioni da tenere in considerazione e che sono state anche soggetto di studi euristici nello sviluppo di Tutela [45], un'applicazione web che fornito un indirizzo Ethereum restituisce una valutazione della sua anonimità, sono:

- Evitare di effettuare un deposito da un indirizzo, e il ritiro corrispondente sullo stesso indirizzo.
- Evitare di utilizzare lo stesso prezzo del gas per il pagamento di operazioni di deposito e ritiro.
- Evitare di effettuare transazioni tra l'indirizzo che ha effettuato il deposito e quello che ha effettuato il ritiro.
- Evitare di eseguire una serie di depositi a partire da un indirizzo e poi effettuare una serie di ritiri corrispondente alla serie di depositi (ad esempio: otto depositi da 1 ETH ciascuno provenienti da un indirizzo "x" e otto ritiri da 1 ETH ciascuno su un indirizzo "y").

## Capitolo 5

# Protocollo di risoluzione proposto

In questo capitolo verrà descritta l'implementazione, proposta dallo studio effettuato, per la realizzazione di un sistema di votazioni elettronica su blockchain.

La parte più importante dell'architettura sarà quindi costituita dagli smart contract, realizzati tramite Solidity, in particolare ci si soffermerà su quello del token di voto, pool di Tornado Cash e il contratto per effettuare le votazioni.

Sono state poi sviluppate delle applicazioni che permettono al votante di interagire con gli smart contract tramite una User Interface. Nella fase di autenticazione, il votante accederà ad una pagina web contenente un form nella quale dovrà inserire i suoi dati personali e un indirizzo Ethereum su cui desidera ricevere il token di voto (vedere Figura 5.1).

### Form autenticazione

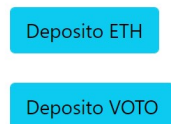
<b>Nome:</b>	<input type="text" value="Nome"/>
<b>Cognome:</b>	<input type="text" value="Cognome"/>
<b>Email:</b>	<input type="text" value="Email"/>
<b>Numero telefono:</b>	<input type="text" value="xxxxxxxx"/>
<b>Indirizzo Ethereum:</b>	<input type="text" value="0xx"/>
<input type="button" value="Invia"/>	

Figura 5.1: Applicazione web per autenticazione votante

Per effettuare il deposito del token di voto e ETH negli smart contract pool, il votante premerà i pulsanti nella pagina dedicata, che si occuperà di effettuare le

operazioni in automatico (come riportato in Figura 5.2 e in Figura 5.3).

## Deposito su smart contract pool



Deposito ETH

Deposito VOTO

Figura 5.2: Applicazione web per deposito su smart contract pool

## Ritiro da smart contract pool



Nota:

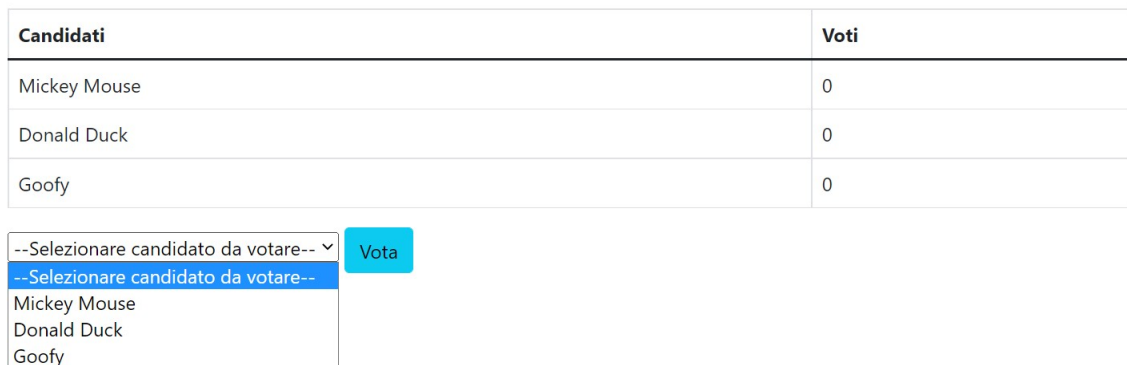
Indirizzo Ethereum:

Ritiro

Figura 5.3: Applicazione web per ritiro da smart contract pool

Per poter votare l'utente dovrà selezionare il candidato desiderato dal menu a tendina della pagina apposita, premendo poi il pulsante "Vota" il token di voto verrà inviato allo smart contract, che registrerà il voto (vedi Figura 5.4).

## Votazioni



Candidati	Voti
Mickey Mouse	0
Donald Duck	0
Goofy	0

--Selezionare candidato da votare-- ▾

--Selezionare candidato da votare--

Mickey Mouse

Donald Duck

Goofy

Vota

Figura 5.4: Applicazione web per effettuare il voto

Nella Tabella 5.1 è presente un elenco di abbreviazioni, usate all'interno di questo capitolo.

Abbreviazione	Spiegazione
ACCOUNT1	Account Ethereum con ether ed eventualmente altri token, quindi non anonimo, riconducibile al votante
ACCOUNT2	Account Ethereum senza nessuna transazione e perciò non riconducibile in nessun modo all'identità del votante
OV	Organizzatore delle votazioni, o chi per lui
SCP	Smart contract pool

Tabella 5.1: Abbreviazioni

## 5.1 Architettura del progetto

A seguire verrà esposta l'architettura del progetto, descrivendo in che modo interagiscono applicazioni web e smart contract, indicando i vari linguaggi di programmazione usati (vedi Figura 5.5).

Nella fase iniziale di autenticazione, il votante accederà ad una pagina web realizzata in HTML connessa ad un database MySQL tramite PHP, in caso di corretta autenticazione, tramite l'uso di JavaScript, verrà chiamata automaticamente la funzione dello smart contract del token di voto che conierà un nuovo token e lo invierà ad un indirizzo Ethereum scelto dal votante.

Sono poi presenti due pagine web, realizzate in HTML e PHP, che usano JavaScript per interagire con lo smart contract pool e consentono quindi il deposito e ritiro del token di voto e di ETH .

Infine tramite una pagina web in HTML che utilizza la libreria JavaScript “ethers.js” si realizzerà una dApp che permetterà al votante di accedere al suo wallet MetaMask e poi di votare inviando il suo token di voto allo smart contract dedicato alle votazioni.

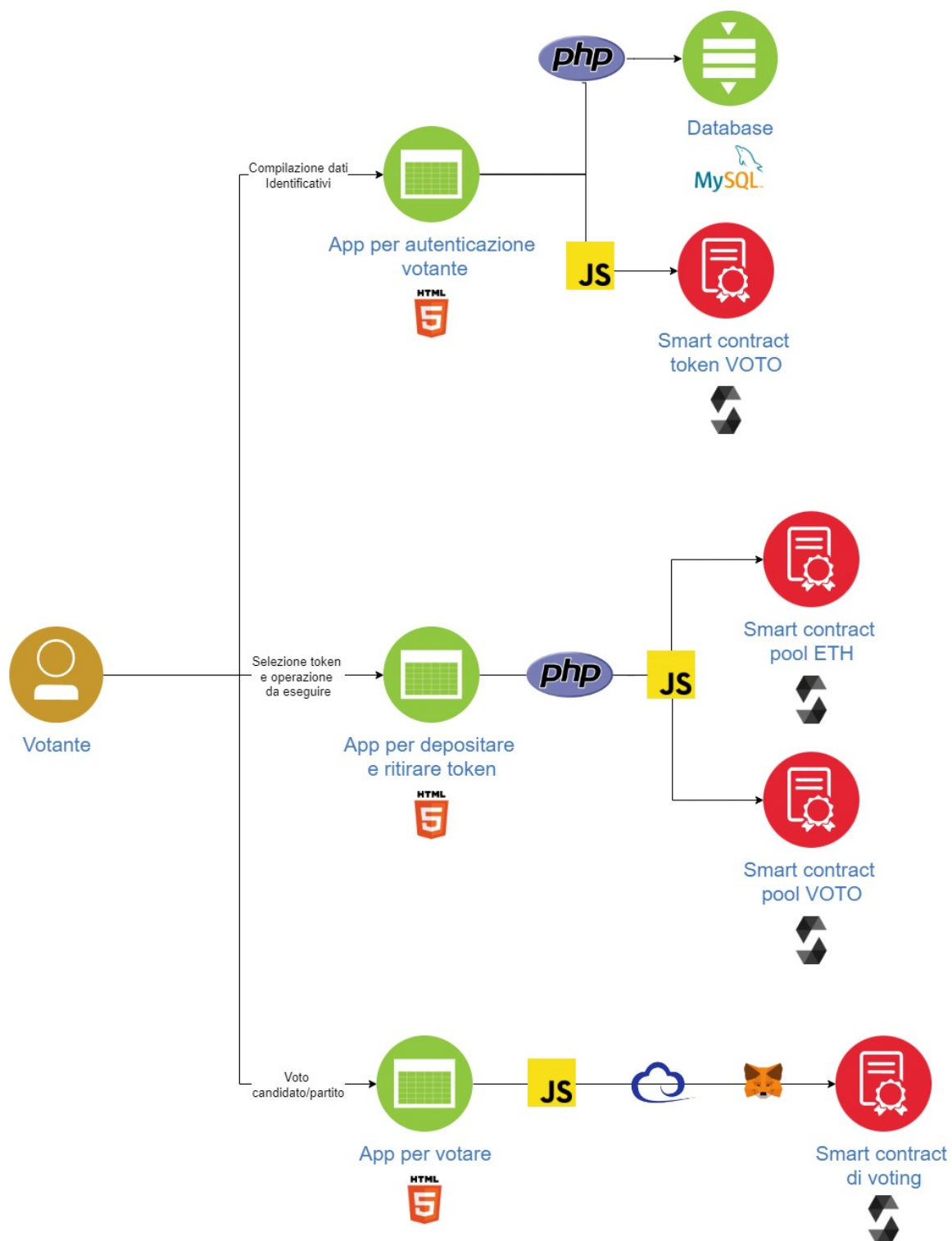


Figura 5.5: Architettura del progetto

Facendo riferimento all'architettura del progetto, di seguito verrà descritto il protocollo di risoluzione proposto da questo studio (come vedremo in Figura 5.6). Per prima cosa l'organizzatore delle votazioni (OV) effettuerà il rilascio dello smart contract di un token ERC20, questo token avrà la sigla "VOTO", l'utilità del token sarà quella di dimostrare la corretta identificazione dei votanti, a questi verrà infatti distribuito un token VOTO una volta si siano identificati, l'altra funzione del token sarà quella di permettere di esprimere un voto inviando il token allo smart contract che si occupa delle votazioni.

Verrà fissata una determinata data entro la quale gli utenti che vogliono votare si dovranno identificare (e quindi ricevere 1 VOTO) e fare il deposito di 0.0015 ETH (attualmente equivalenti a circa € 1.5<sup>1</sup>) e 1 VOTO nei corrispettivi smart contract pool (SCP), facendo uso di una ulteriore applicazione web; senza però effettuare il ritiro di nessuno dei due token prima della data prefissata, in modo tale da non compromettere la propria anonimità.

Una volta superata questa data gli utenti autorizzati potranno procedere ad effettuare i ritiri dei token depositati:

- ritiro tramite relay di 0.0015 ETH su un nuovo indirizzo Ethereum (ACCOUNT2), con nessuna transazione in entrata o uscita, e quindi non riconducibile alla propria identità;
- ritiro di 1 VOTO tramite l'indirizzo sul quale è stato effettuato il ritiro di 0.0015 ETH.

Finalmente per effettuare l'operazione di voto, si dovrà accedere alla dApp apposita connettendo il proprio wallet MetaMask e inviando il token VOTO allo smart contract dedicato alle votazioni.

---

<sup>1</sup>La data di controllo è 22/06/2022. <https://bitflyer.com/en-eu/ethereum-chart>

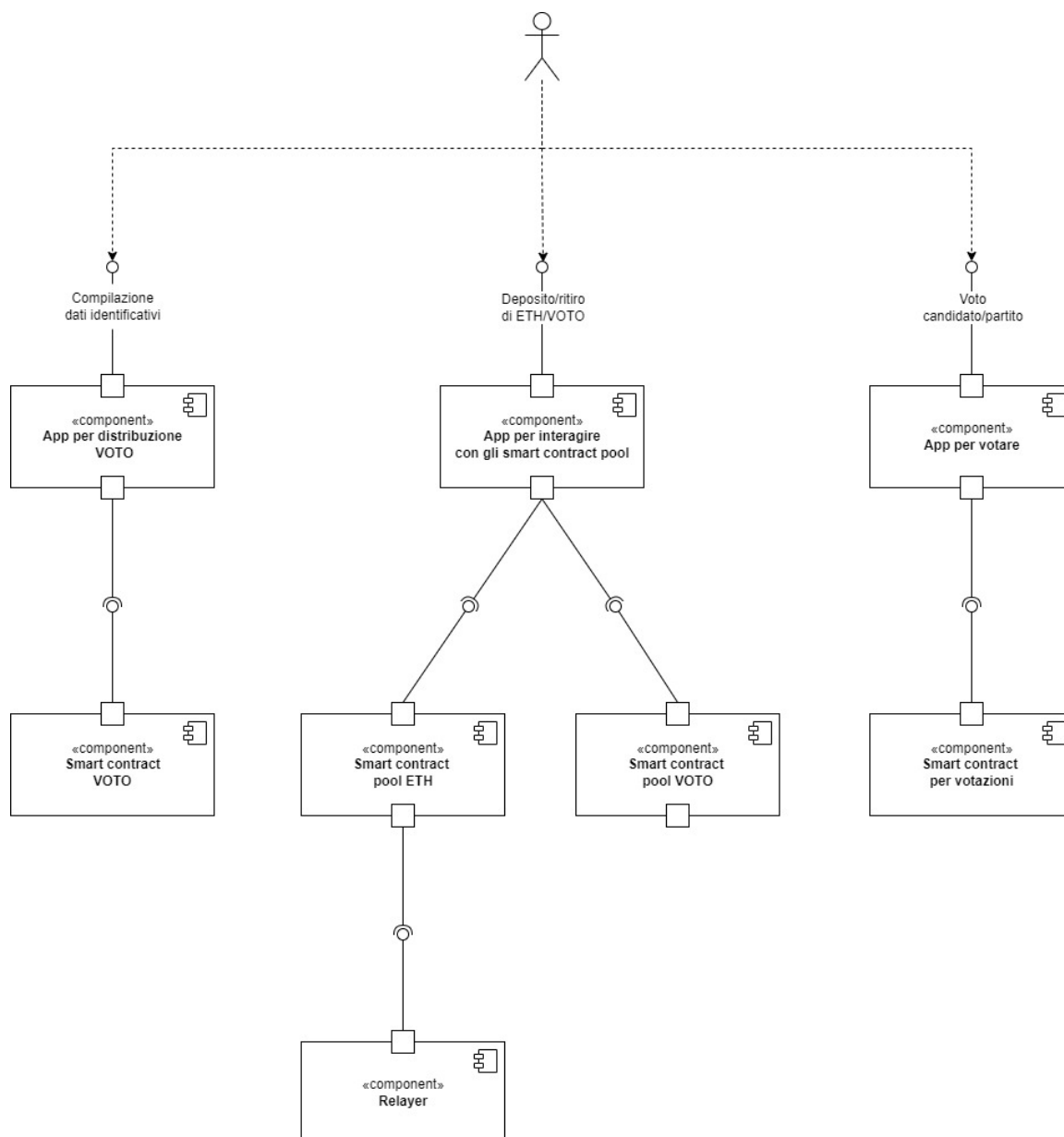


Figura 5.6: Diagramma dei casi componenti: Architettura del progetto

Nelle successive sezioni verranno espone le soluzioni agli obiettivi definiti per implementare il protocollo di e-voting (come riportato in Sezione 1.1):

1. Distribuzione di un token dedicato alle votazioni.
2. Anonimizzazione dell'utente.



3. Rilascio dApp dedicata alle votazioni.

## 5.2 Soluzione Obiettivo 1. Distribuzione di un token dedicato alle votazioni

Nel framework del progetto un token VOTO verrà assegnato ad un utente, autorizzato dall'OV, che si sarà identificato con successo e permetterà a questo di poter esprimere il suo voto consumando il suo token VOTO. L'OV dovrà effettuare l'implementazione dello smart contract di un nuovo token ERC20 (VOTO), che verrà utilizzato per garantire il diritto di voto. Questo verrà rilasciato su Goerli, un network di testing Ethereum, tramite Hardhat, un ambiente di sviluppo Ethereum.

Lo smart contract del token VOTO sarà di proprietà dell'account Ethereum che lo ha rilasciato (OV), e solo questo avrà l'autorizzazione per coniare e distribuire nuovi token (come riportato in Listato 5.1).

```
pragma solidity ^0.8.0;
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract VotoToken is ERC20, Ownable{
    constructor() ERC20("VotoToken", "VOTO"){
        function issueToken(address receiver) public onlyOwner{
            _mint(receiver, 1*10**18);
        }
    }
}
```

Listato 5.1: Smart contract VOTO Token

Infine verrà creata una pagina web contenente un form, che l'utente che vuole ottenere il token VOTO, dovrà compilare con i propri dati personali quali: nome, cognome, email e numero di telefono. Questi dati verranno confrontati con quelli presenti su un database, e nel caso ci sia corrispondenza con una tupla, e questa non sia stata già utilizzata precedentemente per ottenere un token, allora un token VOTO verrà distribuito ad un indirizzo Ethereum scelto dall'utente votante (vedi Figura 5.7).

Data la sua natura decentralizzata, si potrà quindi controllare dallo smart contract il numero di token in circolazione e gli account Ethereum proprietari di essi.

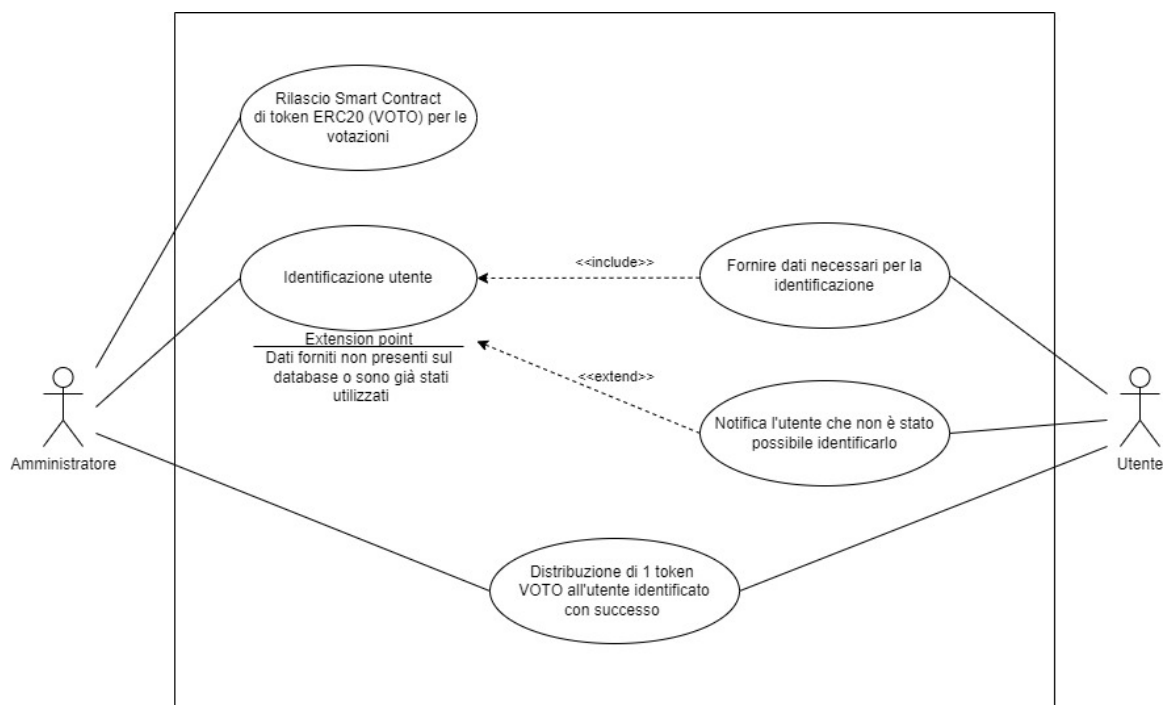


Figura 5.7: Diagramma dei casi d'uso: Distribuzione di un token dedicato alle votazioni

### 5.3 Soluzione Obiettivo 2. Anonimizzazione dell'utente

Per poter garantire il trasferimento in modo anonimo di token VOTO e ETH dall'account Ethereum contenente il token VOTO ad un nuovo account Ethereum (ACCOUNT2), si farà riferimento al protocollo implementato da Tornado Cash (vedi Capitolo 4). L'OV dovrà procedere con il rilascio di SCP per il deposito e ritiro di ETH e VOTO. Questo SCP permetterà di effettuare depositi a partire da ACCOUNT1 per poi effettuare ritiri su ACCOUNT2. Gli utenti che avranno ottenuto il token VOTO e desiderano votare, per poterlo fare anonimamente, dovranno quindi prima compiere questi due passi: deposito e ritiro tramite SCP di ETH e VOTO.

L'utente quindi che avrà effettuato il deposito, al successo della transazione, riceverà in output una nota privata che dovrà necessariamente memorizzare per poter successivamente ritirare l'importo depositato.

Come già è stato descritto l'OV stabilirà una data di scadenza per effettuare i depositi, una volta superata questa data il votante effettuerà prima il ritiro di ETH su un ACCOUNT2, ma per pagare la fee di questa transazione non utilizzerà l'ACCOUNT1 (se lo facesse si creerebbe una correlazione tra l'ACCOUNT1 e l'ACCOUNT2, e quindi si perderebbe l'anonimità) nè l'ACCOUNT2 (questo non possiede nessun ether). Per pagare la transazione del ritiro di ETH verrà quindi fatto uso di un account Ethereum detto relayer, configurato dall'OV.

Effettuata questa operazione l'utente avrà a disposizione nel suo ACCOUNT2 0.0015 ETH che in parte verranno usati per effettuare il ritiro del token VOTO dallo SCP e successivamente chiamare le funzioni dello smart contract di voto.

Per poter rendere possibile la procedura descritta, allora l'OV dovrà implementare:

- Uno SCP per il deposito e ritiro di ETH e uno SCP per il deposito e ritiro di VOTO.

Per rilasciare gli SCP si è fatto riferimento alla repository *tornado-core*<sup>2</sup> di Tornado Cash su GitHub. Sono quindi state fatte delle modifiche per inserire la chiave privata dell'indirizzo che si vuole rilasci lo SCP, e per scegliere il token e l'importo che accetterà lo SCP.

Effettuata questa configurazione iniziale si potrà procedere al rilascio dello SCP, per fare ciò verranno chiamate da riga di comando delle istruzioni.

Nel caso di ETH verrà chiamata la seguente istruzione:

```
npx truffle migrate --network goerli --reset --f 2 --  
to 4
```

L'istruzione chiamata per l'ETH SCP effettuerà il rilascio (vedi Figura 5.8):

- dello smart contract hasher, che si occuperà di calcolare il Pedersen Hash (vedere Sezione 4.6) quando verrà effettuato un deposito, tramite l'esecuzione di *2\_deploy\_hasher.js*

---

<sup>2</sup><https://github.com/tornadocash/tornado-core>

- dello smart contract verifier, che si occuperà di verificare che la prova (come riportato in Sezione 4.5) quando verrà effettuato un ritiro sia valida, tramite l'esecuzione di *3\_deploy\_verifier.js*
- dell'ETH SCP tramite l'esecuzione di *4\_deploy\_eth\_tornado.js*

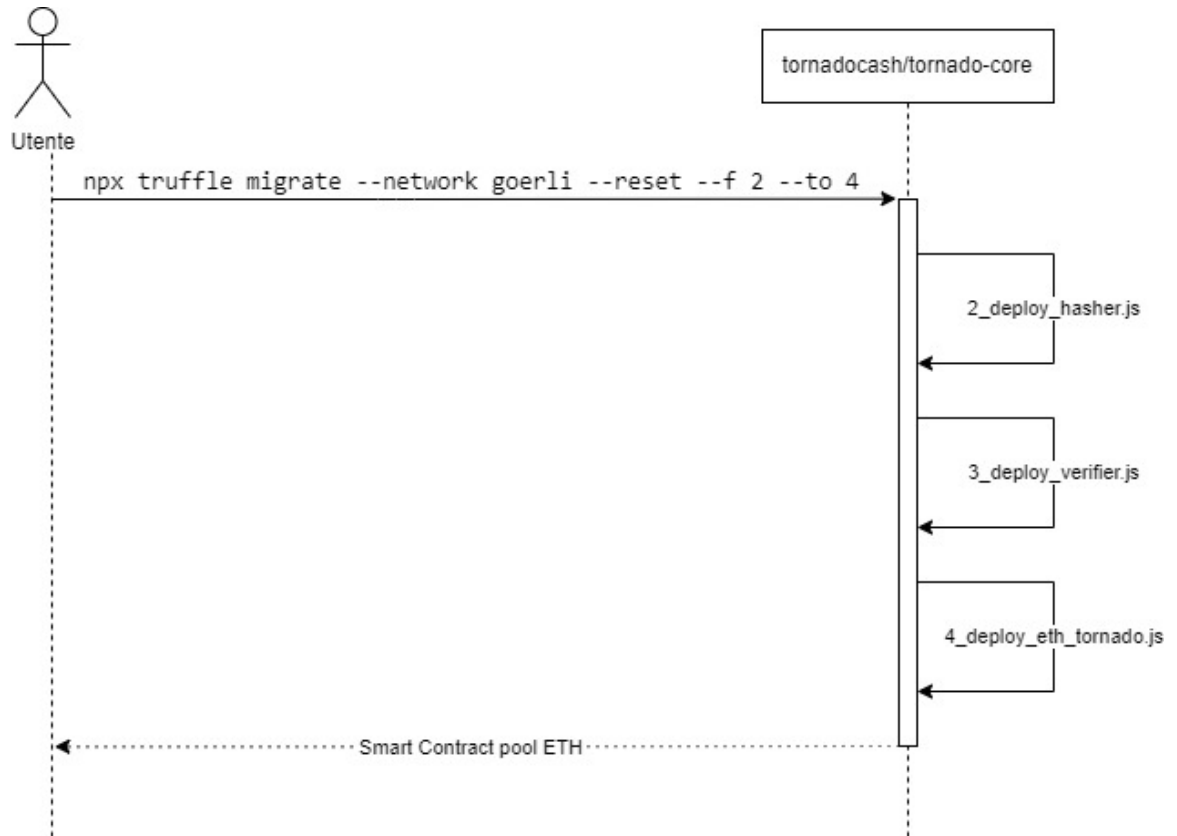


Figura 5.8: Diagramma di sequenza: Rilascio SCP ETH

Nel caso di VOTO verranno chiamate queste due istruzioni:

```
npx truffle migrate --network goerli --reset --f 2 --to 3
```

```
npx truffle migrate --network goerli --reset --f 5
```

Le istruzioni chiamate per il VOTO SCP effettueranno il rilascio (vedi Figura 5.9):

- dell'hasher tramite l'esecuzione di `2_deploy_hasher.js`.
- del verifier tramite l'esecuzione di `3_deploy_verifier.js`.
- del VOTO SCP tramite l'esecuzione di `5_deploy_erc20_tornado.js`.

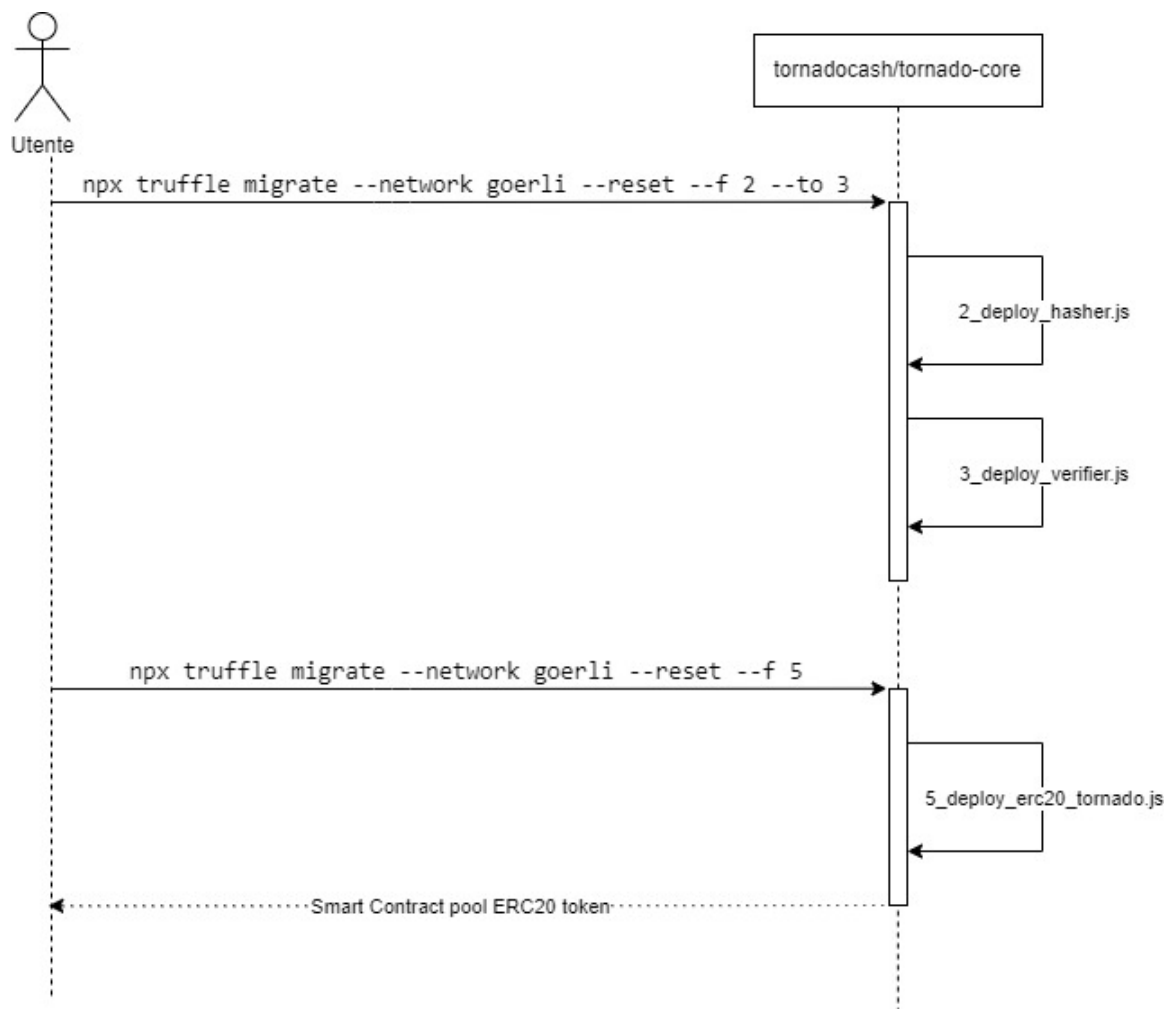


Figura 5.9: Diagramma di sequenza: Rilascio SCP ERC20 token

- Un *relayer*: account Ethereum che permetterà di ritirare in modo anonimo i token ETH.

Per configurare il relayer si è fatto riferimento alla repository *tornado-relayer*<sup>3</sup> di Tornado Cash su GitHub.

<sup>3</sup><https://github.com/tornadocash/tornado-relayer/tree/c838316436a9f87f8655087c34764b46e4b1491b>

Alla configurazione già presente è stata aggiunta la possibilità di operare su Goerli e con gli SCP già rilasciati di ETH e VOTO, modificare il valore del gas utilizzato nelle transazioni ed il valore delle fee ricevute dal relayer nel caso di una operazione di ritiro da uno SCP che lo coinvolga.

La configurazione del relayer, sopra descritta, sarà sviluppata su un ambiente Docker tramite Docker Compose. Ciò permetterà al relayer di essere operativo fintanto che il container Docker sarà in esecuzione.

- Uno strumento da riga di comando e una applicazione web che permettano l'interazione con gli SCP.

L'implementazione di queste applicazioni fanno riferimento alla repository *tornado-cli*<sup>4</sup> di Tornado Cash su GitHub.

Così come per il relayer è stata aggiunta la possibilità di modificare il valore del gas utilizzato nelle transazioni, operare su Goerli e con gli SCP già rilasciati di ETH e VOTO.

Di seguito verrà elencata la procedura che dovrà eseguire l'utente per interagire con gli SCP, completa di comandi e grafi esplicativi (vedi Figura 5.10 e Figura 5.11).

L'utente, come è stato descritto, dovrà effettuare un deposito di 0.0015 ETH dall'ACCOUNT1 verso lo SCP, per fare ciò verrà chiamato da riga di comando l'istruzione:

```
node cli.js deposit ETH 0.0015 --rpc https://goerli.
infura.io/v3/9aa3d95b3bc440fa88ea12eaa4456161
```

Questa istruzione chiamerà, dall' ETH SCP, la funzione di deposito:

```
deposit(bytes32 key)
```

Dove il parametro *key* sarà il commitment che verrà inviato all'importo depositato all'ETH SCP.

---

<sup>4</sup><https://github.com/tornadocash/tornado-cli/tree/2e4d59aed06e2c58f79e2cbe521b80b87cdeb067>

Se il deposito verrà effettuato con successo, all'utente verrà restituita una nota che dovrà memorizzare per poi poter effettuare il ritiro. Un esempio di nota che verrebbe restituita nel caso di chiamata di questa istruzione è:

```
tornado-eth-0.0015-5-0x7085039a0e7be80208540514293817
63be12f25129cf686cbd2e97a6e1f8a3c10458b53ca856d0fb9fe
4cde2d91364709bc7fb29a4d6c88c484524ec4a38
```

Dopodiché verrà eseguita la stessa procedura per il deposito di 1 VOTO dall'ACCOUNT1 verso lo SCP tramite l'istruzione:

```
node cli.js deposit VOTO 1 --rpc https://goerli.
infura.io/v3/9aa3d95b3bc440fa88ea12eaa4456161
```

La nota che si otterrà sarà del tipo:

```
tornado-dvt-1-5-0xad61ef02e188c1fcd36f2e6df409449417d
383014ccd4fad12bb46926ca9bcf2a3f62e3954fc09468d87d9bf
c298b7835f1dc23b612ce621154d38e97653
```

Si procederà poi ad effettuare il ritiro di ETH tramite il relay con destinatario l'indirizzo dell'ACCOUNT2.

Sia l'indirizzo dell'ACCOUNT2:

```
0x73F2e73BFDF3DdeB7C16C300787f957B36506329
```

L'istruzione per il ritiro sarà:

```
node cli.js withdraw
tornado-eth-0.0015-5-0x7085039a0e7be80208540514293817
63be12f25129cf686cbd2e97a6e1f8a3c10458b53ca856d0fb9fe
4cde2d91364709bc7fb29a4d6c88c484524ec4a38
0x73F2e73BFDF3DdeB7C16C300787f957B36506329
--rpc https://goerli.infura.io/v3/
9aa3d95b3bc440fa88ea12eaa4456161
--relay http://relaybruno.duckdns.org
```

Questa istruzione chiamerà, dall'ETH SCP, la funzione di ritiro:

```
withdraw(bytes _proof, bytes32 _root, bytes32
        _nullifierHash, address _recipient, address _relayer,
        uint256 _fee, uint256 _refund)
```

Nella funzione verrà indicato l'indirizzo del destinatario (parametro *\_recipient*), una commissione di transazione (parametro *\_fee*), e tramite una prova, si dimostrerà di possedere un segreto al quale corrisponde un commitment memorizzato nella lista dello smart contract, il quale però non sia già stato utilizzato per fare un ritiro. Questa prova (parametro *\_proof*) però non dovrà rivelare il deposito corrispondente al segreto, viene quindi utilizzato il protocollo *zero-knowledge proof*.

Verrà poi selezionata una radice (parametro *\_root*) tra quelle memorizzate sul contratto e si calcolerà l'hash del nullifier (parametro *\_nullifierHash*).

Lo smart contract, una volta verificata la prova e l'unicità dell'hash del nullifier, procederà a trasferire *N* token all'indirizzo del destinatario.

Si eseguirà poi il ritiro di VOTO usando l'ACCOUNT2, tramite la seguente istruzione:

```
node cli.js withdraw
tornado-dvt-1-5-0xad61ef02e188c1fcd36f2e6df409449417d
383014ccd4fad12bb46926ca9bcf2a3f62e3954fc09468d87d9bf
c298b7835f1dc23b612ce621154d38e9765
0x73F2e73BFDF3DdeB7C16C300787f957B36506329
—rpc https://goerli.infura.io/v3/
9aa3d95b3bc440fa88ea12eaa4456161
```



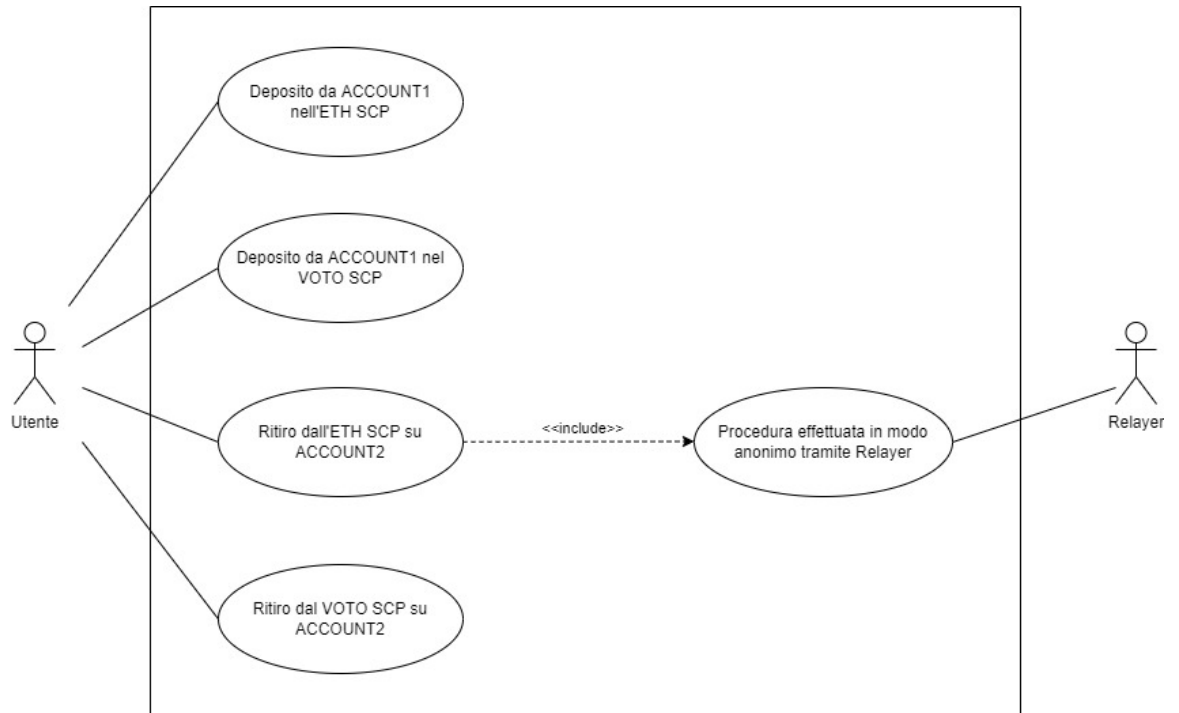


Figura 5.10: Diagramma dei casi d'uso: Interazione con SCP

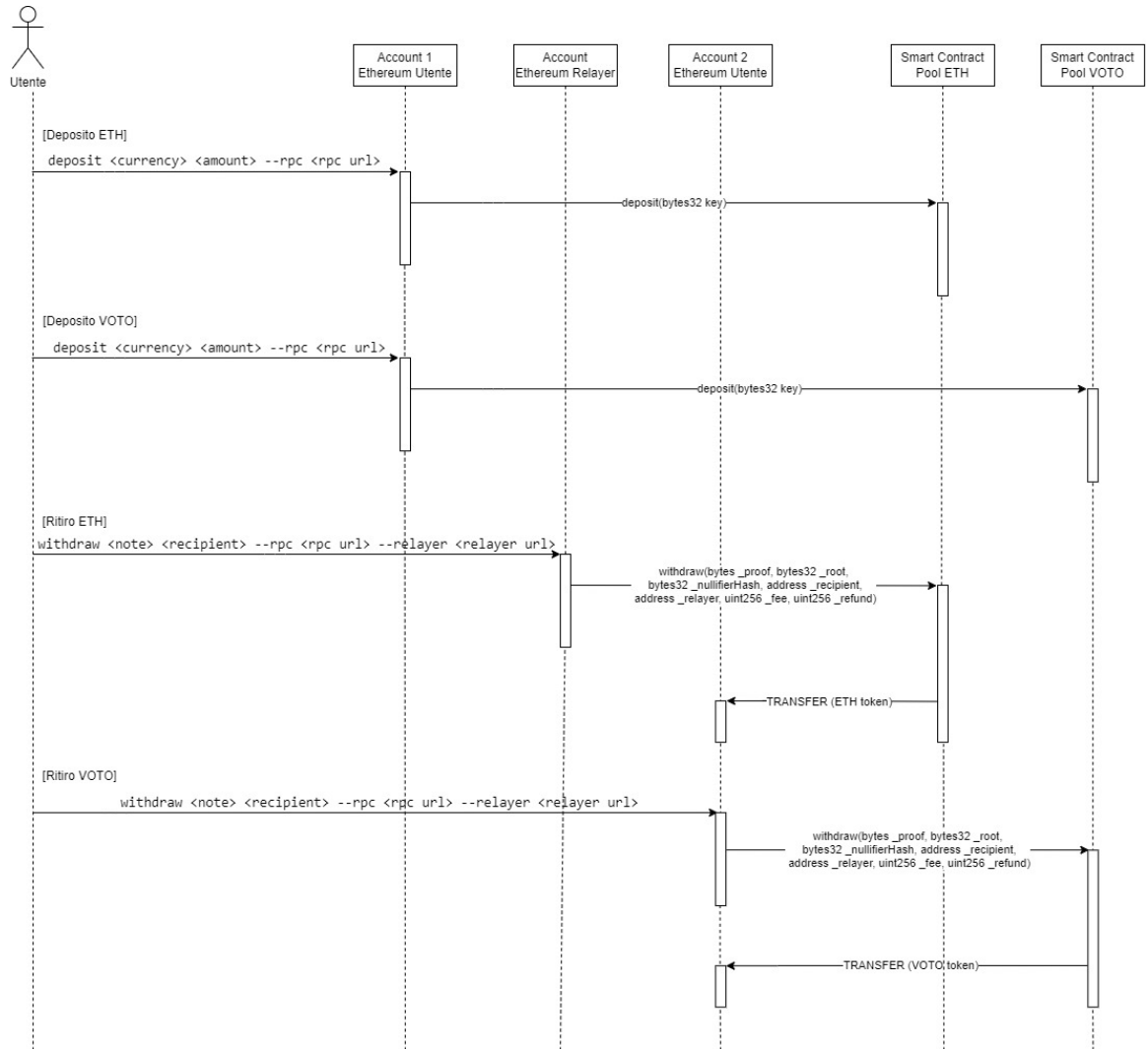


Figura 5.11: Diagramma di sequenza: Interazione con SCP

## 5.4 Soluzione Obiettivo 3. Rilascio dApp dedicata alle votazioni

Se l'utente avrà seguito correttamente le procedure descritte nei capitoli precedenti (come riportato nel Capitolo 5.2 e nel Capitolo 5.3) si ritroverà un indirizzo Ethereum non riconducibile alla sua identità (ACCOUNT2) con ETH per pagare le fee delle transazioni e VOTO per poter votare.

L'OV dovrà allora implementare una web dApp che permetta all'utente di votare. Alla base dell'applicazione ci sarà quindi uno smart contract che permetterà di impostare i candidati, il token e la quantità da accettare per poter votare. Questi parametri verranno configurati tramite il costruttore dello smart contract:

```
constructor(string[] memory proposalNames, uint256  
tokenQuantita_, address token)
```

- il parametro *proposalNames* creerà un array dinamico che conterrà i nomi dei candidati alle votazioni;
- il parametro *tokenQuantita\_* imposterà la quantità di token necessari per ciascun utente affinché possa votare;
- il parametro *token* indicherà l'indirizzo del token ERC20 che verrà richiesto per poter votare.

La funzione che si occuperà delle votazioni nello smart contract sarà:

```
vote(uint proposal)
```

- il parametro *proposal* indicherà il candidato selezionato dall'utente votante.

Una volta rilasciato lo smart contract, verrà sviluppata una applicazione web che interagisca con quest'ultimo (come vedremo in Figura 5.12).

La procedura ideata prevede che il votante effettui l'accesso sul suo account MetaMask, in cui avrà configurato il suo ACCOUNT2, dall'estensione su browser<sup>5</sup>. All'avvio dell'applicazione web dedicata alle votazioni verrà chiamata automaticamente la funzione JavaScript del frontend:

```
getVotes()
```

Questa funzione si occuperà di prendere i nomi e il numero di voti dei candidati, dallo smart contract, e mostrarli sulla pagina web. Chiamerà poi la funzione dello smart contract del token VOTO:

```
approve(address spender, uint256 amount)
```

---

<sup>5</sup><https://metamask.io/download/>

Mediante questa funzione MetaMask notificherà all'utente la richiesta di esecuzione di una transazione che consentirà allo smart contract dedicato alle votazioni (*spender*) di ricevere 1 (*amount*) VOTO da parte del wallet dell'utente: nel caso questo token sia presente nel suo wallet e nel caso l'utente proceda con la votazione.

L'utente dovrà poi aspettare qualche secondo affinché venga effettuato il mining di questa transazione sulla blockchain, una volta fatto ciò l'utente potrà selezionare il candidato che desidera votare dall'apposito menu a tendina dalla pagina web e poi cliccare sul pulsante con la scritta "Vota".

Cliccato il pulsante verrà chiamata la funzione JavaScript:

```
vote()
```

Questa funzione si occuperà di chiamare la funzione dello smart contract di voto:

```
vote(uint proposal)
```

Tale che *proposal* costituirà il candidato selezionato dal votante. A sua volta questa funzione chiamerà due funzioni dello smart contract del token VOTO:

- `allowance(address owner, address spender)`

Sarà la prima funzione a venire chiamata, e controllerà che l'utente abbia concesso allo smart contract di voto (*spender*) di spendere un token VOTO dal suo wallet (*owner*).

- `transferFrom(address sender, address recipient, uint256 amount)`

In caso la chiamata alla prima funzione abbia avuto esito positivo, si cercherà di fare la chiamata della seconda funzione e quindi MetaMask chiederà all'utente la conferma per l'esecuzione della transazione, se questa verrà concessa si invierà 1 (*amount*) VOTO dal votante (*sender*) allo smart contract di voto (*recipient*) ed infine verrà registrata con successo l'operazione di voto.

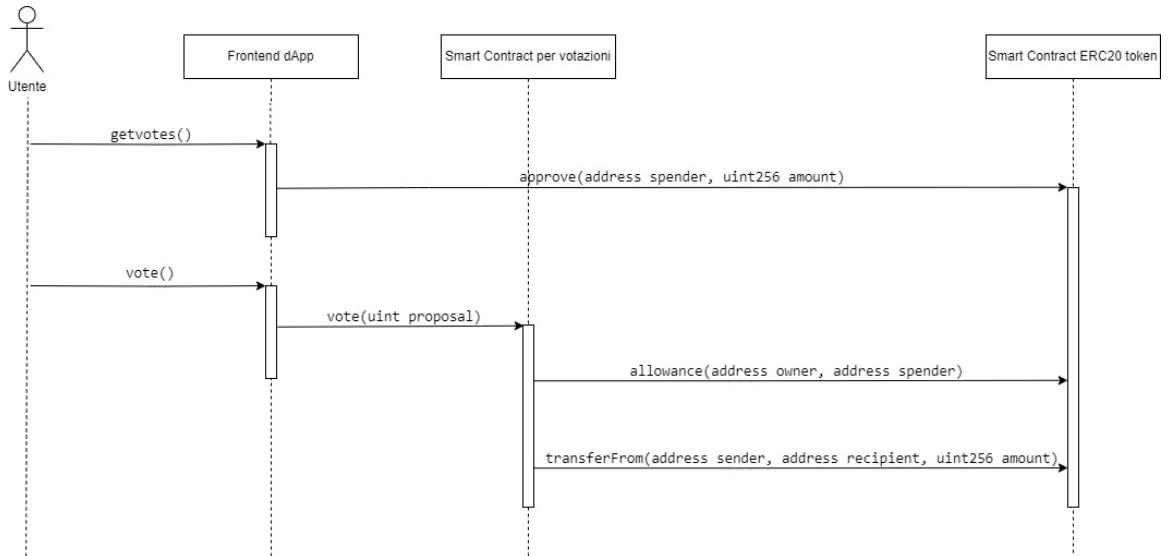


Figura 5.12: Diagramma di sequenza: Voto da parte dell'utente

## 5.5 Stima dei costi

Per calcolare il costo medio della fee di ciascun tipo di transazione sono stati presi in analisi i valori riportati durante una fase di test dello sviluppo del progetto durata all'incirca un mese.

Di seguito verranno elencati i costi delle fee di tutte le transazioni che dovrà compiere un utente che vuole votare:

- Il deposito di 0.0015 ETH sullo SCP<sup>6</sup> avrà una fee dal costo medio di 0.0010805 ETH sull'ACCOUNT1.

`deposit(bytes32 key)`

- Il deposito di 1 VOTO sullo SCP<sup>7</sup> avrà una fee dal costo medio di 0.001291 ETH sull'ACCOUNT1.

`deposit(bytes32 key)`

<sup>6</sup><https://goerli.etherscan.io/address/0xedb857246146300271bde0dbe01a87eb9000ea1d>

<sup>7</sup><https://goerli.etherscan.io/txs?a=0x797e1f8a3b00ddde4e4319ce946d85905cdb9ef0&p=1>

- Il ritiro di 1 VOTO dallo SCP<sup>8</sup> avrà una fee dal costo medio di 0.000448857 ETH sull'ACCOUNT2.

```
withdraw(bytes _proof, bytes32 _root, bytes32
        _nullifierHash, address _recipient, address _relayer,
        uint256 _fee, uint256 _refund)
```

- Il permesso allo smart contract di voto di spendere 1 VOTO<sup>9</sup> dal wallet del votante avrà una fee dal costo medio di 0.000063152 ETH sull'ACCOUNT2.

```
approve(address spender, uint256 tokens)
```

- La votazione tramite l'invio di 1 VOTO allo smart contract di voto<sup>10</sup> avrà una fee dal costo medio di 0.000107894 ETH sull'ACCOUNT2.

```
vote(uint256 proposal)
```

L'ACCOUNT1 pagherà mediamente in fee 0.0023715 ETH, mentre l'ACCOUNT2 pagherà in fee 0.000512009 ETH.

Quindi complessivamente l'importo medio che verrà speso in fee è di 0.0028835 ETH.

---

<sup>8</sup><https://goerli.etherscan.io/txs?a=0x797e1f8a3b00ddde4e4319ce946d85905cdb9ef0&p=1>

<sup>9</sup><https://goerli.etherscan.io/txs?a=0x26d50f9cf1e4c6a1a09e2c56391ec78d7efc311b>

<sup>10</sup><https://goerli.etherscan.io/address/0x0e1278def1606fb85ce2827409489a2a89ef21d3>

# Capitolo 6

## Conclusioni

In questo studio è stato descritto il processo di realizzazione di un sistema di votazione elettronica basato sulla blockchain Ethereum, utilizzando il protocollo implementato dal coin mixer Tornado Cash per garantire la privacy del votante e tramite la tecnologia blockchain e gli smart contract per garantire la trasparenza dell'intero processo. Di seguito verrà riassunta la realizzazione dello studio portato a termine e verranno descritte le conclusioni tratte al raggiungimento dei risultati aspettati dagli obiettivi posti dallo studio (vedere Sezione 1.1).

In primo luogo è stata creata una applicazione web che autentichi il votante e a seguito di ciò distribuisca un token VOTO, che sarà poi impiegato nelle fasi successive del protocollo per permettere all'utente di esprimere il proprio voto.

Successivamente facendo riferimento al protocollo di Tornado Cash sono stati rilasciati degli smart contract pool che permettono il deposito di token VOTO e di ETH, e che al successo dell'operazione restituiscono una nota privata che permetterà poi di effettuare il ritiro in maniera anonima su un indirizzo non riconducibile alla propria identità. Per permettere al votante di interagire con i sopracitati smart contract è stata poi sviluppata una applicazione web che in maniera semplice e intuitiva consentirà l'operazione di deposito e ritiro. Per quanto riguarda il ritiro di ETH è stato effettuato il settaggio di un account relayer che si occuperà di pagare le fee di questo tipo di transazione preservando l'anonimità dell'indirizzo che riceverà il ritiro.

In fine è stata sviluppata una dApp che consente al votante di effettuare l'operazione di voto interagendo con lo smart contract che si occupi della votazione.

A seguire verrà analizzato in che modo il protocollo studiato soddisfi le proprietà

tipiche di un sistema di voto elettronico (come riportato in Sezione 2.4).

L'*unicità* di voto sarà derivata dal fatto che lo smart contract che si occupa delle votazioni registrerà il voto una volta che riceverà la chiamata della funzione apposita insieme ad un token VOTO; ciascun votante riceverà un solo token VOTO dopo la corretta identificazione e una volta inviato allo smart contract delle votazioni non sarà poi possibile recuperarlo in alcun modo.

La struttura a blocchi della blockchain, dove ogni blocco è collegato al precedente tramite degli hash, renderà la modifica o l'eliminazione di una transazione impossibile da non rilevare e richiederebbe una elevata potenza computazionale per portare a termine l'attacco, garantendo quindi l'*integrità* del sistema.

L'*attendibilità*, così come l'integrità, dipenderà per una buona parte dall'infrastruttura usata, in questo caso Ethereum, una delle blockchain più sicure e popolari.

La blockchain Ethereum avrà un registro delle transazioni pubblico ed accessibile a chiunque, si potrà quindi *verificare* la corretta registrazione del proprio voto e il corretto conteggio di tutti i voti registrati.

Tramite una semplice funzione dello smart contract sarà possibile effettuare il *conteggio* dei voti registrati, ed essendo questi memorizzati in dei blocchi nella blockchain, come descritto precedentemente non sarà possibile modificarli e quindi si potrà ripetere il conteggio in maniera sicura.

Attraverso l'applicazione web implementata sarà possibile per ciascun votante *autenticarsi* e al successo di questa operazione ricevere il token VOTO.

L'utilizzo del protocollo di Tornado Cash garantirà la *privacy* tramite il trasferimento di ETH e VOTO su un indirizzo Ethereum non riconducibile all'identità del votante.

Possibili sviluppi futuri del protocollo descritto che possono migliorare quelle che sono le funzionalità già presenti sono:

- implementare una dApp, che consenta all'utente di accedere al proprio account Ethereum tramite MetaMask e mediante questo poi possa interagire con gli smart contract pool effettuando depositi e ritiri in maniera sicura e decentralizzata;
- garantire le proprietà di *confidenzialità* e *assenza di evidenza* nel sistema di e-voting, attraverso una implementazione del progetto che faccia utilizzo di



una permissioned blockchain dove il diritto di leggere all'interno della blockchain venga concesso solo a determinati utenti, in modo tale che questi possano effettuare il conteggio dei voti una volta finite le votazioni [11];

- integrare l'architettura disegnata con un protocollo che permetta una corretta e sicura autenticazione del votante:
  - autorizzazione tramite una one-time password (OTP);
  - autenticazione tramite riconoscimento biometrico;
  - autenticazione a più fattori (MFA);
  - uso di protocolli OAuth e OpenID.

Come si è già spiegato la forza della tecnologia blockchain sta nella distribuzione del registro di transazioni tra i nodi della rete, ai quali aspetterà il compito di verificare e memorizzare i blocchi nel sistema.

Questo tipo di infrastruttura risulta una valida alternativa per varie applicazioni nel mondo reale di trasmissione di dati, tra cui anche quelle di voto. Precedenti applicazioni di voto online sono risultate fallimentari a causa di gravi falle di sicurezza. La blockchain può risolvere questo tipo di vulnerabilità: grazie alla crittografia e decentralizzazione il registro della blockchain sarà incorruttibile e ogni transazione sarà facile da verificare.

I casi d'uso di un sistema di e-voting su blockchain sono diversi, è possibile applicarli a elezioni, sondaggi, censimenti, votazioni aziendali...

Sono sempre di più le compagnie ed organizzazioni che si stanno mobilitando per implementare delle applicazioni di e-voting nel mondo reale. Per esempio nel 2018 in Thailandia il partito democratico ha tenuto delle elezioni per eleggere il proprio leader di partito tramite l'uso di ZCoin, queste sono state le prime elezioni politiche eseguite su larga scala a fare uso della tecnologia blockchain [37].

# Bibliografia

- [1] What are zk-SNARKs? <https://z.cash/technology/zksnarks/>, 2022. Online; consultato il 27 aprile 2022.
- [2] Cosmas Krisna Adiputra, Rikard Hjort, and Hiroyuki Sato. A proposal of blockchain-based electronic voting system. In *2018 second world conference on smart trends in systems, security and sustainability (WorldS4)*, pages 22–27. IEEE, 2018.
- [3] Shubhani Aggarwal and Neeraj Kumar. Chapter fifteen - blockchain 2.0: Smart contractsworking model. In Shubhani Aggarwal, Neeraj Kumar, and Pethuru Raj, editors, *The Blockchain Technology for Secure and Smart Applications across Industry Verticals*, volume 121 of *Advances in Computers*, pages 301–322. Elsevier, 2021. doi: <https://doi.org/10.1016/bs.adcom.2020.08.015>. URL <https://www.sciencedirect.com/science/article/pii/S006524582030070X>.
- [4] Martin Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 191–219. Springer, 2016.
- [5] Andreas M Antonopoulos and Gavin Wood. *Mastering ethereum: building smart contracts and dapps*. O’reilly Media, 2018.
- [6] UN General Assembly et al. Universal declaration of human rights. *UN General Assembly*, 302(2):14–25, 1948.

- [7] Ahmed Ben Ayed. A conceptual secure blockchain-based electronic voting system. *International Journal of Network Security & Its Applications*, 9(3):01–09, 2017.
- [8] ayefda (pseudonimo). Introduction to Tornado Cash. <https://docs.tornado.cash/general/readme>, 2022. Online; consultato il 27 aprile 2022.
- [9] ayefda (pseudonimo). How does Tornado Cash work? <https://docs.tornado.cash/general/how-does-tornado.cash-work>, 2022. Online; consultato il 27 aprile 2022.
- [10] Jordi Baylina and Marta Bellés. 4-bit window pedersen hash on the baby jubjub elliptic curve.
- [11] Stefano Bistarelli, Ivan Mercanti, Paolo Santancini, and Francesco Santini. End-to-end voting with non-permissioned and permissioned ledgers. *Journal of grid computing*, 17(1):97–118, 2019.
- [12] bt11ba (pseudonimo). Tips to remain anonymous. <https://docs.tornado.cash/general/tips-to-remain-anonymous>, 2022. Online; consultato il 27 aprile 2022.
- [13] Rumeysa Bulut, Alperen Kantarcı, Safa Keskin, and Şerif Bahtiyar. Blockchain-based electronic voting system for elections in turkey. In *2019 4th International Conference on Computer Science and Engineering (UBMK)*, pages 183–188. IEEE, 2019.
- [14] Vitalik Buterin. Ethereum white paper: A next generation smart contract & decentralized application platform. 2013. URL <https://github.com/ethereum/wiki/wiki/White-Paper>.
- [15] Tornado Cash. Introducing Private Transactions On Ethereum NOW! <https://tornado-cash.medium.com/introducing-private-transactions-on-ethereum-now-42ee915babe0>, 2019. Online; consultato il 27 aprile 2022.
- [16] Tornado Cash. How to stay anonymous with Tornado.cash and similar solutions. <https://tornado-cash.medium.com/how-to-stay-anonymous-with-torna>

- do-cash-and-similar-solutions-efdecdbd7d37, 2020. Online; consultato il 27 aprile 2022.
- [17] Alessandra Cerioni and Federica Cenacchi. Aspetti di sicurezza nelle elezioni elettroniche. 2003.
  - [18] Benoît Chevallier-Mames, Pierre-Alain Fouque, David Pointcheval, Julien Stern, and Jacques Traoré. On some incompatible properties of voting schemes. In *Towards Trustworthy Elections*, pages 191–199. Springer, 2010.
  - [19] Corwin Smith. Transactions. <https://github.com/ethereum/ethereum-org-website/blob/dev/src/content/developers/docs/transactions/index.md>, 2022. Online; consultato il 10 luglio 2022.
  - [20] Gaby G Dagher, Praneeth Babu Marella, Matea Milojkovic, and Jordan Mohler. Broncovote: Secure voting system using ethereum’s blockchain. 2018.
  - [21] Michael Dunn and Laurence Merkle. Overview of software security issues in direct-recording electronic voting machines. In *Proceedings of the ICCWS 2018 13th International Conference on Cyber Warfare and Security, Washington, DC, USA*, pages 8–9, 2018.
  - [22] Francesco Fusco, Maria Ilaria Lunesu, Filippo Eros Pani, and Andrea Pinna. Crypto-voting, a blockchain based e-voting system. In *KMIS*, pages 221–225, 2018.
  - [23] Raghavendra Ganji and BN Yatish. Electronic voting system using blockchain, 2018.
  - [24] Friðrik Þ 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.
  - [25] Markus Jakobsson and Ari Juels. *Proofs of Work and Bread Pudding Protocols(Extended Abstract)*, pages 258–272. Springer US, Boston, MA, 1999. ISBN 978-0-387-35568-9. doi: 10.1007/978-0-387-35568-9\_18. URL [https://doi.org/10.1007/978-0-387-35568-9\\_18](https://doi.org/10.1007/978-0-387-35568-9_18).

- [26] Anuj Kharel. Analysis of security properties of electronic voting scheme and possible vulnerabilities, 05 2020.
- [27] Yi Liu and Qi Wang. An e-voting protocol based on blockchain. *Cryptology ePrint Archive*, 2017.
- [28] Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*, pages 369–378. Springer, 1987.
- [29] Terry Newman. Tasmania and the secret ballot. *Australian Journal of Politics & History*, 49(1):93–101, 2003. doi: <https://doi.org/10.1111/1467-8497.00283>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8497.00283>.
- [30] Nishant Sachdeva. Introduction to Tornado Cash. <https://github.com/ethereum/solidity/blob/v0.8.15/docs/index.rst>, 2022. Online; consultato il 20 luglio 2022.
- [31] Harsha V Patil, Kanchan G Rathi, and Malati V Tribhuwan. A study on decentralized e-voting system using blockchain technology. *Int. Res. J. Eng. Technol*, 5(11):48–53, 2018.
- [32] Paul Wackerow. Intro to Ethereum. <https://github.com/ethereum/ethereum-org-website/blob/dev/src/content/developers/docs/intro-to-ethereum/index.md>, 2022. Online; consultato il 10 luglio 2022.
- [33] Paul Wackerow. Introduction to dapps. <https://github.com/ethereum/ethereum-org-website/blob/dev/src/content/developers/docs/dapps/index.md>, 2022. Online; consultato il 10 luglio 2022.
- [34] Michał Pawlak, Aneta Poniszewska-Marańda, and Natalia Kryvinska. Towards the intelligent agents for blockchain e-voting system. *Procedia Computer Science*, 141:239–246, 2018.
- [35] Alexey Pertsev, Roman Semenov, and Roman Storm. Tornado cash privacy solution version 1.4. , 2019.

- [36] Péter Szilágyi. Clique PoA protocol Rinkeby PoA testnet. <https://github.com/ethereum/EIPs/issues/225>, 2022. Online; consultato il 31 luglio 2022.
- [37] Reuben Yap. World’s First Large-Scale Blockchain-Based Political Election Held on Zcoin’s Blockchain. <https://firo.org/2018/11/13/worlds-first-large-scale-blockchain-based-political-election-held-on-zcoins-blockchain.html>, 2022. Online; consultato il 31 luglio 2022.
- [38] Roy G Saltman. *Effective use of computing technology in vote-tallying*, volume 13. US Department of Commerce, National Bureau of Standards, 1978.
- [39] Sam Richards. Introduction to smart contracts. <https://github.com/ethereum/ethereum-org-website/blob/dev/src/content/developers/docs/smart-contracts/index.md>, 2022. Online; consultato il 10 luglio 2022.
- [40] Sam Richards. Networks. <https://github.com/ethereum/ethereum-org-website/blob/dev/src/content/developers/docs/networks/index.md>, 2022. Online; consultato il 22 luglio 2022.
- [41] Ashish Singh and Kakali Chatterjee. Secevs: Secure electronic voting system using blockchain technology. In *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*, pages 863–867. IEEE, 2018.
- [42] SK Vivek, RS Yashank, Yashas Prashanth, N Yashas, and M Namratha. E-voting systems using blockchain: An exploratory literature survey. In *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 890–895. IEEE, 2020.
- [43] Fabian Vogelsteller and Vitalik Buterin. Eip 20: Erc-20 token standard. *Ethereum Improvement Proposals*, 20, 2015.
- [44] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.
- [45] Mike Wu, Will McTighe, Kaili Wang, Istvan A Seres, Nick Bax, Manuel Puebla, Mariano Mendez, Federico Carrone, Tomás De Mattey, Herman O Demaestri, et al. Tutela: An open-source tool for assessing user-privacy on ethereum and tornado cash. *arXiv preprint arXiv:2201.06811*, 2022.

- [46] Haibo Yi. Securing e-voting based on blockchain in p2p network. *EURASIP Journal on Wireless Communications and Networking*, 2019(1):1–9, 2019.
- [47] Bin Yu, Joseph K Liu, Amin Sakzad, Surya Nepal, Ron Steinfeld, Paul Rimba, and Man Ho Au. Platform-independent secure blockchain-based voting system. In *International Conference on Information Security*, pages 369–386. Springer, 2018.