Alma Mater Studiorum · Università di Bologna

Scuola di Ingegneria e Architettura
Dipartimento di Informatica · Scienza e Ingegneria · DISI
Corso di Laurea in Ingegneria Informatica

SISTEMI ANTI-DENIAL OF SERVICE IN AMBIENTI ANONIMI BASATI SU ZK-SNARK

Relatore:
Prof. Paolo Bellavista

Presentata da:
Straccali Leonardo

Anno Accademico 2022/2023

Indice

Elenco delle figure

Capitolo 1

Analisi del protocollo RLN

Qui in una prima fase parlo di cosa è RLN, come nasce da chi nasce, ora sotto chi è e chi lo usa. parlo velocemnte di cosa fa e breve spoiler di come lo fa, poi dico che devo introdurre dei concetti che poi usero RLN (Rate limiting nullfier) is a construct based on zero-knowledge proofs that enables spam prevention mechanism for decentralized, anonymous environments. In anonymous environments, the identity of the entities is unknown.

The anonymity property opens up the possibility for spam attack and sybil attack vectors for certain applications, which could seriously degrade the user experience and the overall functioning of the application.

The RLN construct prevents sybil attacks by increasing the cost of identity replication. To be able to use an application that leverages the RLN construct and be an active participant, the users must provide a stake first. The stake can be of economic or social form, and it should represent something of high value for the user. The user identity replication will be very costly or impossible in some cases (again depending on the application). Providing a stake does not reveal the user's identity, it is just a membership permit for application usage, a requirement for the user to participate in the app's specific activities

The proof system of the RLN contract enforces revealing the user credentials upon breaking the anti-spam rules. By having the user's credentials, anyone can remove the user from the application and withdraw their stake. The user credentials are associated with the user's stake

1.1 Strumenti

1.1.1 Merkel tree

Quindi abbiamo bisogno di una struttura dati dove inserire in nostri dati sara più chiaro cosa e come la usiamo nella pricedura di registrazione am ad ognni modo possiamo inizia re a paralrne essenzialmentei i merkel tree sono degli alberi solitamente binari ma possono avere arietà diverse che sono composti nelle fogli da hash di segnali dati e da le non fogli che sono gli hash dei composti delli figli fino ad arrivare

alla radiche che è un uncio hash che rappresta tuto l'albero, qusto tipo di struttra dati viene usato ep due motivi principali, la possibilità di verificare molto velocemnte se sono avvenuuti cambiamenti e di campire relativamente velocemente quali, e soprattul apossibilità di gerare velocemnte con colessita nei temi O(lon n) dove n è il nuemro di folgi dell'albero se un elemento appariene o no, le due proprietà insime permmetndo i ottenre una struttura dati che occpa poco spaizo e che può generare quelle che vengono chaiamte proof of membershi p molto velocemntea per generare una prova quello che bisogna fare èfornire gli hash che servono per tootenre la root quindi immaginando che peggi voglia dimostrare a victor di esssre nell'albero senza rivelagli la sua identitha potra forntire immaginando che la'alero sia il seguent una prova contenete come com paramentri pubblici gli ha sh dei nosti della'albero che sono utili ad arrivare alla radice in questo caso adfkjladjfjadlkfjas, e cpo paramnteo segretoi l'ash del primo elemnto ovvero la foglia, la prova testimonierò che il segreto è effiteviamente l'ahs della prima foglia e che se si ripercorrono i livelli ahshando in modo correto si raggiunge un rispltato ovvero una radice ora al verificatore basterà controllare che la radice che p in suo posseeso sia uguale alla radice pasata. se un malfattore provase a manomettere gli elemnti dell'albero per provare a convicere un dimostraore allora la root cambierebbe e per la proitàa di non collisone delgi ahash non dovrebe riuscirci.

in realtà si mette come inpuit pubblicogli stemp di hash e come primato il io segreto , e sarà lui a calcoalre

io ho merkel tree che è fatto in un determinato modo poi ho Incremental Merkle Tree Problem per l'inderimento e ho anche Merkel proof per la proof of membership.

1.1.2 Nuove funzioni di hash

qui parlo di poseidon dico perchè si usa, ovvero in generale perchè bisomga usare nuove hash funciton il motivo è il fatto di lavorare in in filed che nei circuiti algebrci gli hash normali sono dispendiosi e lenti mebtr quest fatti a posta sono veloce sia in termini di tempo che di vincoli del circuito r1cs infatti faccio vedere differenze tra sha e poseidon, inoltre poseido è peferito anche rispetto ai suoi amichetti.

uno dei mattoni base della crittografia in generale sono le funzioni hash e le ZKP non fanno a meno di questi utilissi strumenti iffati come abbiamo visto anche in precedenzanella sezionedi zk-snark abbiamo accennato a prticolarei funzioni di hashgi per poter elaborare informazioni senza condividerle. Per produrre i circui zk-snar quindi si fa spesso uso dell funzionoin shi has il problema che usndo funzioni di shash solite com e l'hash256 che non stono state precgttate conm lo scopo di zk-snarksi si ottengono circuito dove i costrinti cioè i vincoli doveti alle fiunzioni hash sono molto elevati il che rende le funzioni hash il punto di bottiglia di prove si in temrini di tempo che di e ne aumenta anche di molto la memtoira per ovviare a questo problema negli hani sono state costruite dell funzioni hash appostie che lavorando nativamente sul dominoi dei campi finiti e applciando diverse stratefie di mixing sono in grado di onttenre glo setssi risultati in temini di sicurezza ma con molti meno vincoli è più velocitàa, negli ultimi tempi la funzione hash più utlizzata è poseidon hashh che è stata pubblcaita nel 2021 e presenta prestazioni molto allettanti, nelle tabelle sotto possimao vedre in confronto gli algoritimi che usano poseindo controo

i sui compettore e contro o SHA256

SHA256 is a cryptographic hash function that uses a set of well-defined steps to transform the input message into a fixed-length output of 256 bits. The SHA256 algorithm involves several rounds of bit-wise operations that transform the input message in a way that is designed to be difficult to reverse. The algorithm is designed to be secure against a wide range of attacks, including collisions and pre-image attacks.

Poseidon, on the other hand, is a permutation-based hash function that is designed to be efficient and resistant to certain types of attacks. Poseidon uses a set of round functions that apply a series of bit-wise operations to the input message, similar to SHA256. However, unlike SHA256, the number of rounds and the specific operations used in each round are customizable. This allows Poseidon to be tailored to specific applications and hardware platforms.

One key difference between SHA256 and Poseidon is that SHA256 is based on the Merkle–Damgård construction, which involves padding the input message to a fixed length before processing it. This can introduce certain vulnerabilities, such as length-extension attacks. Poseidon, on the other hand, is a sponge function that does not require padding and is designed to be more resistant to such attacks.

1.1.3 Nullifier

Qui in questa sezione mi piacerebbe dire cos è da un punto di vista intuitivo, e perchè mi è utule: i nullifiere sono oggeni che nullificano ovvero sono oggetti crittografici che vengono legati a un messaggio o a un qualcosa in modo da renderli univocamente identificabili ma non in quanto permettono di risalire a chi appartengono ma li rendono unici per la rihciesta transazione mesagio e così via questo permmette loro di risolvere il problema del "duble signaling" mantenendo comunquel aprivacy il problema del double signaling si rende palese conun esempio pensiamo al voto se volessimo costruire un sistema zk-snark per una votazione online sarebbe immediaot pensare che l'anoniamo sia solo una cosa positiva ma come dettto precedentemente mantendendo l'anonimato potremmo esporci a problemi come ad esemoioche uno che vota voti due votle come impediamo qusta cosea possiamo farlo attraverso ul nullifier nello specifico bastera inserire l'identià dell'utne allinterno del merkel tree tramite un commitment che non è altro che un segreto conun hash e creare un nullifier per quel segreto ovvero un altro hash diverso ma sempre legato al segretom metre il coomitemnt p pubblico e visibile tutti il nullifier verrà ostrato solo quando si votera in questo modo la prova potra controllare che tu sei legittmato a votare tramite la proof-of-membrsip e che il nullifier che mi passi è effitavemente relazionato, una volta fatto questo prendo il tuo nullifie re lo metto inuna atrlo merke treec dove ci sono quelli spesi, ora se uno provase ha votare con lo stesso segreto il nullifier sarebbe construito allo stesso modo come il commitment eqindi potremmo ignorare il voto. Questo passagio doverebbe suocare falimaile se posso evitare che una pesona mi sciriv du evolte perche non basta il concetto d luffiere per implementare un rate -limiting il concetto di base pè che con il nullifier pui renderti conot se qualcuno ha usato los tesso sgreto per pubblicare o fare una irhicesta ma non hai idea di cquesto sia in quanfo la prova di tpermette di saere solo che il nullifier è già satto usato e

da uno del gruppo dei membti ma non chi e leagot a cosa.

n cryptography, a nullifier is a value that is used to invalidate or "nullify" a previous commitment to a certain value or secret. n these systems, nullifiers are used to prevent double-spending or other forms of fraud. When a party makes a commitment to a certain value or secret, they generate a corresponding nullifier. If they later attempt to use the same value in another commitment, the nullifier will reveal that the value has already been used, and the commitment will be considered invalid.

we have to simultaneously solve privacy and the double spending problem

1.2 Registrazione

Before registering to the application the user needs to generate a secret key and derive an identity commitment from the secret key using the Poseidon hash function (identityCommitment = posseidonHash(secretKey)). prima di tutto l'utente si registra per registrarsi devo fornire deve avere una secret key per evitare che sia troppo semplice diventare membro quacosa dobbiamo fare e questo qualcosa è legare la secret key a una form di stake di qualche tipo questo non lo facciamo noi in prima persona anche perchè io non lo volgiio trattare ma ci sono robe tipo interep che lo fanno che forse sarbbe meglio guardare ad ogni modo quando ho finito di ottenre la mia chiave private che se scoperta potrebbe ricondurre come no all amia stake faccio un has per proteggirla in alcuni casi per una protezione più sicura si fanno delle altre robe com per esempio in semaphore o in vac ad ogni modo dopo che hai fatto la registrazione devi inserire il tuo commitment dentro il merkel tree, in abiendete centralizzato questo consiste nel rihiedere onferma di inserimento e che il server inivi la root dell'albero perf ar si che tutti si aggironino

basta invaire la root? come fanno a sapere le sibil leavs mi sa che devi invairgli delle robe in più anche se poche. ad ogni modo dopo chi hai fattoqusta parte hai finito.

The user registers to the application by providing a form of stake and their identity commitment, which is derived from the secret key. The application maintains a Merkle tree data structure (in the latest iteration of the RLN construct we use the Incremental Merkle Tree algorithm), which stores the identity commitments of the registered users. Upon successful registration the user's identity commitment is stored in a leaf of the Merkle tree and an index is given to them, representing their position in the tree

1.3 Interazione

ora viene il bello per interagire devi sicuramente provare di essere parte del merkel tree e questo lo fai con le fogli che servono a provalo e con il tuo identy commitment, ma poi devi anche fornire delle shares inffatti il rate limiting si basa su SSS che è un algoritmo di MPC che permette di dividereun segrto su molteplici share e asicurare che finche n su m dove nz m share sono al sicuro allora anche il segrreto è al sicuro il numero n di share che viene selezionato dipemde dal polinomio ch si sceglie per impemntare ala sss infatti in base al polinomio inffi scegliendo un poliniomio di

grado ncis iassicura che si possano rialscai re n-1sare senza rischio, la valutazione di questo è ababstanza baanel enel caso lenare infatti se immaginiamo un polinimio lineare e inviamo del punti grazie all'interpolazione possimo ricavare il polinimio. Ora otorniamo a noi se creo un poinimio nel seguente modo $p = x * a_1 + a_0$ dove a0 è il segreto e a_1 è un nullifier che cambia in base a epoca e a applicazione allora posso invaire un messaggio senza prolbmi ma se ne inviao due ai ai ai ovvaimente pero devo provare che sto invaito effettivamente share oveero punti x y dove x è il mesagio e y è la valutazione del polinimio al messagio appartenetni al poliniomi altriimenti tuto crollerebbe per questa prova e per la prova di appatenzenza si una zksnark. quando invio un messagiuo allore devo invare le seguenti robe ecccc

1.4 Punizione

quando invio un messagio allora posso scoprire chi è lo stronzo e eliminarlo ma per richiamare quale strunzo è che ha fatto la roba potrei andare in modo empirico ma unso l'identernal didentifier per rivelare a_0 adesso se a_0 è relazionatea ha una froma di stak ela posso requisire ad ogni mdo posso ricavare il polinomial commitment e rimuoverlo da i gichi

OK dal primo articolo letto ho dei dubbi 1come fai a dare la stake e a far si che quando scopri la mia private key tu rialsci la stake, io mi immaginio che quando trovi la mia private key puo togliere il moi commitment dall'albero perchè sono individuabile ma la stake? forse la private key deve essere assocaita alla stake da un diverso sistema bho? Additionally, depending on the application the $identity_secret_hash$ can be used for taking the user's provided stake. questo mi fa pensare che in qualche modo siano legate