Architettura modulare e Crittografia di Solana

Martin Tomassi

24 maggio 2025

Indice

Capitolo 1

Introduzione a Solana

Solana nasce con l'obiettivo di superare i limiti delle generazioni precedenti, in termini di scalabilità, velocità di esecuzione e costi di transazione. Il crescente utilizzo delle principali blockchain, come Bitcoin ed Ethereum, ha evidenziato alcuni problemi strutturali: tempi di validazione elevati, commissioni in crescita e limitazioni nel numero di transazioni al secondo. Questi vincoli hanno incentivato la ricerca di nuove architetture in grado di mantenere l'immutabilità del registro distribuito, ma, allo stesso tempo, offrire throughput e latenza minori rispetto alle precedenti. Le soluzioni di terza generazione, tra cui Cardano, Polkadot e appunto Solana, si distinguono per l'adozione di protocolli ibridi, progettati per massimizzare parallelismo, risposta in tempo reale e modularità del consenso.

Capitolo 2

Panoramica dell'architettura modulare di Solana

Solana si distingue per la sua architettura modulare, dove ogni componente gioca un ruolo cruciale nel raggiungere un throughput elevato. I moduli di questa architettura sono:

2.1 Proof of History (PoH)

Il Proof of History è un "orologio" crittografico globale e decentralizzato che si occupa di registrare, in modo verificabile, il tempo e l'ordine degli eventi all'interno della blockchain. Questa innovazione riduce drasticamente la necessità di comunicazioni e scambi di messaggi tra i nodi per raggiungere il consenso sull'ordine temporale delle transazioni. Il PoH permette ai validatori di concordare sull'ordine degli eventi senza dover comunicare attivamente tra loro per ogni timestamp, migliorando notevolmente l'efficienza e la velocità della rete.

2.2 Tower Byzantine Fault Tolerance (Tower BFT)

Tower BFT è un protocollo di consenso ottimizzato per funzionare in combinazione con il Proof of History. Prendendo ispirazione dai principi del Practical Byzantine Fault Tolerance (PBFT), Tower BFT sfrutta il timestamping fornito dal PoH per raggiungere un accordo sui blocchi in maniera rapida ed efficiente. La sua architettura è progettata per tollerare la presenza di nodi malfunzionanti o malevoli all'interno della rete, garantendo la

continuità e l'integrità del ledger. La collaborazione tra PoH e Tower BFT è un elemento fondamentale per garantire la scalabilità di Solana, in quanto il Proof of History fornisce una cronologia verificabile che semplifica e velocizza il processo di voto e finalizzazione del consenso.

2.3 Turbine

Turbine è il protocollo di propagazione dei blocchi, che si ispira a BitTorrent. Il suo compito principale è quello di distribuire i dati dei blocchi in modo efficiente a migliaia di validatori attraverso la rete. Per raggiungere questo obiettivo, Turbine suddivide i blocchi in piccoli pacchetti e li trasmette utilizzando una struttura ad albero multi-livello, riducendo la complessità di comunicazione da lineare a logaritmica rispetto al numero di nodi. Inoltre, incorpora codici di correzione degli errori (come i codici Reed-Solomon) per garantire che i validatori possano ricostruire il blocco completo anche in presenza di perdita di pacchetti, aumentando la robustezza e la resilienza della rete.

2.4 Gulf Stream

Gulf Stream è un protocollo di gestione delle transazioni che rivoluziona il concetto tradizionale di "mempool" nelle blockchain. Gulf Stream elimina questa necessità inoltrando le transazioni direttamente ai futuri leader (validatori che saranno responsabili della creazione dei prossimi blocchi) con un certo anticipo. Questo permette un'esecuzione anticipata delle transazioni ed una significativa riduzione dell'utilizzo della memoria sui nodi. La predizione del leader, resa possibile dal PoH, consente ai client di inviare le transazioni al validatore corretto con largo anticipo, riducendo la latenza di conferma e migliorando l'efficienza complessiva della rete.

2.5 Sealevel

Sealevel è il runtime parallelo per l'esecuzione degli smart contract. A differenza di quelli sequenziali di molte blockchain, Sealevel è progettato per eseguire simultaneamente milioni di transazioni non conflittuali. Questo è reso possibile dalla capacità degli smart contract su Solana di dichiarare in anticipo gli account che andranno a leggere o modificare, consentendo a Sealevel di identificare e parallelizzare le esecuzioni che non presentano dipendenze. Sfruttando appieno l'hardware multi-core e le GPU, Sealevel massi-

mizza la potenza computazionale disponibile, incrementando drasticamente il throughput delle transazioni e la scalabilità delle applicazioni decentralizzate.

2.6 Pipelining

Il Pipelining è un meccanismo di ottimizzazione che divide il processo di validazione delle transazioni in fasi distinte e le assegna a diverse unità di elaborazione hardware e software. Ogni fase della pipeline lavora su un batch differente di transazioni, consentendo di elaborare più operazioni contemporaneamente in stadi diversi. Le fasi includono la ricezione dei pacchetti, la verifica delle firme, lo scheduling delle transazioni, l'esecuzione degli smart contract e l'aggiornamento dello stato.

2.7 Cloudbreak (Accounts database)

Cloudbreak è il sistema di memorizzazione dello stato, un database progettato per scalare orizzontalmente utilizzando SSD ad alte prestazioni. La sua funzione è quella di gestire l'accesso e la modifica degli account, che rappresentano lo stato della blockchain. Cloudbreak è ottimizzato per permettere accessi concorrenti e veloci ai dati degli account, il che è essenziale per supportare l'esecuzione parallela delle transazioni permessa da Sealevel. La sua architettura è progettata per gestire volumi elevati di operazioni I/O, garantendo consistenza e resilienza anche sotto carichi intensi.

2.8 Archivers

Gli Archivers sono un sistema di storage distribuito e decentralizzato per la storia completa del ledger di Solana. Invece di sovraccaricare i validatori per la conservazione dell'intera storia della blockchain, questo incarico viene delegato a nodi specializzati. Gli Archivers scaricano i dati dal leader e li replicano in modo ridondante, utilizzando tecniche come l'Erasure Coding per garantire la disponibilità a lungo termine e la verifica dell'integrità dei dati storici. Questo approccio garantisce che la storia del ledger rimanga accessibile e verificabile in modo decentralizzato ed efficiente, migliorando la resilienza complessiva della rete.

Capitolo 3

Analisi dettagliata dei moduli architetturali

Dopo aver introdotto l'architettura di Solana, nei prossimi capitoli parlerò in dettaglio di ciascun componente. L'obiettivo è quello di spiegare in modo approfondito il funzionamento, le specifiche tecniche, il ruolo e le interazioni di ogni modulo con gli altri elementi dell'ecosistema.

3.1 Proof of History (PoH)

Il Proof of History rappresenta un approccio innovativo per la sincronizzazione temporale e la verifica dell'ordine degli eventi su Solana. In questo capitolo analizzo in dettaglio le componenti tecniche alla base della sua implementazione, illustrandone il funzionamento e le modalità di integrazione con altre funzioni crittografiche della rete.

3.1.1 Funzionamento SHA-256 sequenziale

Il cuore del meccanismo PoH risiede nell'applicazione iterativa e sequenziale di una funzione hash crittografica, di solito SHA-256. Questo processo genera una sequenza computazionale continua e unidirezionale, dove l'output di ogni operazione hash diventa l'input per il calcolo successivo. Questa catena di hash fornisce una dimostrazione del tempo trascorso.

- **Unicità del percorso computazionale:** l'iterazione sequenziale di SHA-256 rende impossibile "saltare" passaggi intermedi o parallelizzare il calcolo senza invalidare la sequenza. Ogni passo dipende dal risultato del precedente, creando una prova crittografica verificabile dell'ordine temporale degli eventi.

- **Immutabilità:** ogni hash dipende in modo univoco da tutti quelli precedenti, rendendo ogni modifica retroattiva tecnicamente proibitiva.
- **Efficienza verificabile:** i nodi validatori possono ricalcolare la sequenza hash per verificare l'autenticità e l'ordine temporale degli eventi, senza necessità di accordi esterni.

3.1.2 Integrazione con eventi esterni (transazioni)

Il modulo PoH non opera in isolamento, bensì deve interagire con eventi esterni, quali le transazioni e altre operazioni sulla rete:

- **Timestamping delle transazioni:** ogni evento esterno viene inserito nel flusso temporale attraverso un meccanismo di timestamping, associandolo al punto esatto della sequenza hash. Questo permette una cronologia verificabile degli eventi.
- **Sincronizzazione degli input esterni:** la sequenza PoH raccoglie e integra gli input provenienti dalle transazioni garantendo che l'ordine stabilito dalla funzione hash sia mantenuto.
- **Verifica e validazione:** i nodi possono controllare che gli eventi esterni siano conformi all'ordine determinato dalla sequenza PoH, migliorando così la coerenza e l'affidabilità dell'intera rete.

L'integrazione di eventi esterni nel flusso PoH è fondamentale per mantenere una visione globale e coerente degli eventi, permettendo ad ogni transazione di essere verificata rispetto al contesto temporale predeterminato.

3.1.3 Schema di firma per binding temporale

PoH utilizza uno schema di firma digitale che lega ogni evento ad un determinato istante temporale:

- **Autenticità degli eventi:** la firma digitale, associata ad ogni hash, permette di attestare l'origine dell'evento e di verificarne l'integrità.
- **Binding crittografico:** attraverso la firma, il messaggio che costituisce un evento viene "crittograficamente" legato all'hash dell'iterazione precedente, garantendo l'univocità del legame temporale.
- **Non ripudiabilità:** l'utilizzo di chiavi crittografiche asimmetriche assicura che l'autore di un evento non possa negare in un secondo momento di averlo firmato, rafforzando l'affidabilità del sistema.

Questo schema di firma risulta determinante per la creazione di una cronologia ininterrotta e verificabile degli eventi, dove ogni blocco di dati porta con sé la prova crittografica che ne garantisce l'ordine e l'autenticità.

3.1.4 Sincronizzazione, Leadership e Tolleranza ai guasti

É importante capire come PoH contribuisce alla sincronizzazione della rete, al processo di leadership ed alla resilienza complessiva:

- **Sincronizzazione decentralizzata:** sebbene PoH generi un orologio verificabile all'interno del leader, la sua integrazione nel protocollo di consenso (Tower BFT) permette di estendere questa sincronizzazione a tutta la rete. I validatori verificano la sequenza PoH generata dal leader per accertare l'ordine degli eventi proposti nel blocco. Questa verifica, indipendente da parte di ogni validatore, assicura una visione coerente del tempo attraverso la rete, senza la necessità di una dipendenza da una singola fonte di clock esterno.

- **Ruolo del leader nella generazione PoH:** un leader viene eletto per un periodo (slot) specifico per generare la sequenza PoH e proporre un nuovo blocco di transazioni. Il leader esegue continuamente SHA-256, incorporando periodicamente gli hash delle transazioni nel flusso. La lunghezza della sequenza PoH generata, durante quello slot, fornisce una misura del tempo trascorso. Al termine, il testimone della leadership passa al validatore successivo, che continua a generare la propria sequenza PoH a partire dall'ultimo stato verificato. Questa rotazione impedisce la centralizzazione della generazione della sequenza PoH.

- **Tolleranza ai guasti e continuità della rete:** anche in presenza di ritardi di rete o partizioni temporanee, i validatori possono fare affidamento sulla sequenza PoH per verificare l'ordine degli eventi una volta che la comunicazione viene ristabilita. Poiché ogni leader genera la propria sequenza PoH, la rete può continuare a progredire anche se alcuni leader falliscono o sono temporaneamente disconnessi. Il protocollo di consenso Tower BFT, ottimizzato per PoH, sfrutta questa cronologia verificabile per raggiungere rapidamente il consenso sui blocchi, anche in condizioni di rete non ottimali. La capacità dei validatori di verificare autonomamente l'ordine degli eventi riduce la dipendenza da messaggi di coordinamento complessi, migliorando la resilienza della rete.

3.2 Tower BFT

Tower BFT rappresenta il meccanismo di consenso ibrido che integra il Proof of History (PoH) con il Proof of Stake (PoS) in un sistema che ricava ispirazione dai principi del Practical Byzantine Fault Tolerance (PBFT). In questo capitolo, analizzerò in dettagli i principi fondamentali alla base di Tower BFT, evidenziandone le caratteristiche chiave, i vantaggi in termini di sicurezza e scalabilità.

3.2.1 Integrazione tra PoH e PBFT

Il meccanismo Tower BFT sfrutta il flusso temporale generato da PoH per fornire una base sicura e verificabile sull'ordine degli eventi, integrando i principi del PBFT per raggiungere il consenso in presenza di possibili nodi malevoli. In particolare:

- **Utilizzo del timestamping crittografico:** il flusso PoH fornisce una sequenza inconfutabile di eventi, che viene utilizzata per evitare duplicazioni e per mantenere l'ordine temporale delle transazioni.
- **Sottoinsieme di validatori:** i validatori, scelti in maniera ponderata sulla base dello stake, eseguono il protocollo PBFT in un contesto temporale garantito dal PoH, che riduce il numero di round di comunicazione necessari per il consenso.
- **Resilienza a comportamenti malevoli:** l'architettura combinata minimizza l'impatto dei nodi compromessi, dato che il meccanismo PoH aggiunge ulteriore complessità alla possibilità di manipolazioni, integrandosi alla perfezione con i controlli PBFT.

3.2.2 Ruolo dei validatori

Il modello di consenso di Solana prevede la partecipazione di validatori il cui potere di voto è proporzionale allo stake detenuto. Questa metodologia introduce diversi vantaggi:

- **Incentivo economico:** gli stakeholder hanno un incentivo diretto a comportarsi correttamente, poiché il valore del loro investimento è legato alla sicurezza dell'intera rete.
- **Resilienza e decentralizzazione:** il sistema premia la partecipazione attiva dei nodi affidabili, riducendo il rischio di centralizzazione del potere decisionale.

- **Mitigazione degli attacchi:** poiché il peso del voto è associato allo stake, eventuali tentativi di attacco devono affrontare una barriera economica elevata, rendendo tali operazioni meno redditizie o addirittura impraticabili.

3.2.3 Meccanismo di Locking dei voti e finalità

Un aspetto peculiare di Tower BFT è il meccanismo di Locking dei voti, che contribuisce significativamente alla rapidità con cui le transazioni raggiungono la finalità. Questo meccanismo sfrutta PoH per ottimizzare il processo di voto:

- **Votazione basata su Slot e PoH:** i validatori votano su specifici slot, che rappresentano intervalli di tempo definiti dalla sequenza PoH. Invece di votare su ogni singolo blocco in modo isolato, i validatori "bloccano" i loro voti per una sequenza continua di slot, a partire dal primo blocco su cui votano.

- **Supermajority e Locking progressivo:** un blocco viene considerato finalizzato quando una supermajority di stake (superiore ai 2/3) ha votato per esso e per tutti i blocchi precedenti in una catena continua. Il meccanismo di locking fa sì che, una volta che un validatore ha votato per un certo blocco, il suo stake è implicitamente impegnato anche per i blocchi successivi, fino a quando non si verifica un evento specifico (come un fork rilevato). questo "locking progressivo" accelera il raggiungimento della finalità, poiché non è necessario un nuovo round completo di votazioni per ogni blocco successivo.

- **Finalità rapida e prevedibile:** Solana è in grado di raggiungere la finalità delle transazioni in tempi relativamente brevi (spesso entro pochi blocchi, tipicamente nell'ordine di centinaia di millisecondi).

- **Mitigazione dei problemi del PBFT:** l'integrazione con PoH aiuta a mitigare alcuni problemi legati al PBFT tradizionale. Ad esempio, la dipendenza da round di comunicazione sincroni può essere un collo di bottiglia in reti con alta latenza. PoH fornisce un ordine temporale verificabile indipendente dalla comunicazione diretta tra i nodi, riducendo la necessità di molteplici round di "pre-prepare", "prepare" e "commit" per ogni blocco. Il flusso temporale di PoH funge da base per coordinare i voti, semplificando il protocollo e migliorando la sua efficienza in ambienti distribuiti.

- **Penalità per voti incoerenti (slashing):** per incentivare un comportamento onesto e prevenire attacchi, Tower BFT implementa meccanismi di penalizzazione (slashing). I validatori che votano in modo incoerente o tentano di "doppia spesa" (votando per blocchi concorrenti nella stessa altezza) rischiano di perdere parte del loro stake.

3.2.4 Analisi degli attacchi al consenso (33% vs 51%)

La sicurezza dei protocolli di consenso si misura anche in base alla loro capacità di resistere ad attacchi di tipo Byzantine, e l'architettura Tower BFT è stata progettata per tollerare una certa percentuale di nodi malevoli:

- **Soglia del 33%:** il sistema rimane sicuro fino a quando meno di un terzo dei nodi sono compromessi. Tale soglia è cruciale per garantire che un numero sufficiente di validatori onesti possa impedire la corruzione del consenso.
- **Soglia del 51%:** un attacco che raggiunga o superi il 51% dello stake totale mette a rischio l'integrità dell'intero sistema, permettendo potenzialmente il controllo della rete.

Un confronto tra queste due soglie evidenzia quanto segue:

1. **Resilienza al 33%:** finché la percentuale di validatori malevoli resta inferiore al 33%, il meccanismo Tower BFT garantisce un'elevata robustezza, grazie ai controlli incrociati e all'ordine temporale definito dal PoH.
2. **Criticità oltre il 51%:** il superamento del 51% dello stake potrebbe determinare un broken-chain attack, in cui il potere di voto concentrato nei validatori malevoli potrebbe alterare l'ordine degli eventi, riscrivere la cronologia o compromettere la finalità delle transazioni. Questo scenario espone la necessità di meccanismi di penalizzazione e di incentivi correttivi per prevenire la centralizzazione di potere in singoli nodi.

3.3 Turbine

Turbine è il protocollo di propagazione dei dati di Solana, progettato per distribuire rapidamente i blocchi su una rete di migliaia di nodi con latenza minima e uso efficiente della banda. In questo capitolo analizzerò in dettaglio le componenti tecniche di Turbine, illustrandone il funzionamento, le ottimizzazioni di affidabilità e il suo contributo alla scalabilità e resilienza dell'intera rete.

3.3.1 Suddivisione in pacchetti e trasmissione a fanout

Il primo passo, nel funzionamento di Turbine, è la frammentazione di ogni blocco di transazioni in pacchetti di dimensione fissa, tipicamente 64 KiB. Questa modularità è essenziale per l'efficienza della propagazione.

- **Chunking dei blocchi:** un blocco completo di transazioni viene suddiviso in più chunk indipendenti. Ogni chunk viene numerato e identificato in modo univoco, permettendo ai validatori di ricostruire il blocco originale anche se i chunk arrivano fuori ordine o da fonti diverse.

- **Albero di fanout:** il leader attuale della rete, dopo aver creato e firmato un nuovo blocco, non lo invia direttamente a tutti i validatori in una singola trasmissione, ma invia ciascun chunk ad un sottoinsieme ristretto di validatori (che risiedono nel "primo livello" dell'albero di fanout) che, a loro volta, ritrasmettono il chunk ad un altro sottoinsieme di validatori (nel "secondo livello"), e così via, creando una struttura ad albero multi-livello. Ogni nodo riceve i dati solo da un numero limitato di peer, distribuendo il carico di banda e computazionale su tutta la rete.

- **Complessità logaritmica:** con un "fanout" di f nodi per livello (cioè, ogni nodo ritrasmette a f nodi), la propagazione completa di un blocco raggiunge N nodi in circa $\log_f N$ salti. Questo riduce drasticamente il numero di connessioni dirette e la latenza complessiva necessaria per distribuire il blocco all'intera rete, rendendo Turbine altamente scalabile anche con decine di migliaia di validatori. Ad esempio, con un fanout di 8, per raggiungere 4096 nodi bastano 4 salti, anziché 4096 connessioni dirette.

3.3.2 Codici di correzione d'errore e robustezza

Per garantire che i validatori ricostruiscano l'intero blocco anche in presenza di perdita di pacchetti, Turbine integra codici di correzione d'errore:

- **Reed-Solomon encoding:** prima della trasmissione, i chunk del blocco vengono codificati utilizzando schemi di codifica Reed-Solomon. Questa è una tecnica di codifica a correzione d'errore che aggiunge dati di parità ridondanti ai chunk originali. La potenza di Reed-Solomon risiede nella sua capacità di permettere la ricostruzione del blocco originale anche se una percentuale significativa dei pacchetti (fino al 33%) è andata perduta durante la trasmissione.

- **Ridondanza controllata:** il leader ha la capacità di inviare pacchetti supplementari o dati di parità aggiuntivi a nodi selezionati, in particolare quelli che potrebbero trovarsi in percorsi di rete più soggetti a perdita di pacchetti. Questa "ridondanza controllata" mira a migliorare ulteriormente l'affidabilità della propagazione nelle aree più critiche della rete.

- **Verifica integrità:** ogni chunk include checksum crittografici che consentono ai validatori di rilevare immediatamente se un pacchetto è stato corrotto durante la trasmissione o se è mancante. Nel caso in cui viene rilevato un errore, il validatore può richiedere selettivamente il pacchetto corrotto o mancante ad altri peer, evitando di dover scartare l'intero blocco e attendere una ritrasmissione completa.

3.3.3 Ottimizzazione della banda e latenza

Turbine ottimizza l'uso della rete e minimizza la latenza di propagazione dei blocchi grazie a:

- **UDP streaming:** Turbine sfrutta UDP, anziché TCP, dato che, essendo un protocollo "connectionless", non richiede handshake per stabilire una connessione né meccanismi di back-off di congestione. Sebbene UDP non garantisca la consegna dei pacchetti o l'ordine, esso riduce significativamente l'overhead di comunicazione e la latenza per la trasmissione di grandi volumi di dati, rendendolo ideale per la propagazione rapida di blocchi frammentati. Le perdite di pacchetti vengono gestite dai codici Reed-Solomon e dalle richieste selettive, anziché dal ritrasmissione automatica e sequenziale di TCP.

- **Adaptive fanout:** il numero di peer per livello può essere adattato dinamicamente in base alla topologia di rete ed alla latenza misurata. In una rete con bassa latenza ed alta connettività, un fanout maggiore può accelerare la propagazione. Al contrario, in condizioni di rete meno ideali, un fanout più conservativo può prevenire il congestionamento.

- **Parallelismo multi-threaded:** i validatori sono progettati per processare e ritrasmettere i pacchetti di Turbine in thread separati. Questo approccio multi-threaded previene la creazione di colli di bottiglia nelle operazioni di I/O. Mentre un thread riceve nuovi pacchetti, un altro può contemporaneamente applicare la decodifica Reed-Solomon e un terzo può preparare i pacchetti per la ritrasmissione. Questo parallelismo interno massimizza l'utilizzo delle risorse della CPU e della banda.

3.3.4 Contributo alla scalabilità e resilienza di rete

L'integrazione di Turbine nel sistema di Solana garantisce:

- **Scalabilità su larga scala:** la complessità $\mathcal{O}(\log N)$, rispetto al numero di nodi, permette di scalare a decine di migliaia di validatori senza che la latenza di propagazione aumenti esponenzialmente.
- **Tolleranza ai guasti di rete:** grazie ai codici di correzione e al fanout multilivello, la rete rimane operativa ed i blocchi possono essere ricostruiti anche in presenza di perdite significative di pacchetti (fino al 33%) o di fallimenti parziali della rete. Questa robustezza è fondamentale per la stabilità e la disponibilità della rete in casi reali ed imprevedibili.
- **Throughput elevato:** riducendo l'overhead di trasmissione (grazie ad UDP) e sfruttando in modo efficiente la banda, Turbine contribuisce in modo diretto a sostenere il throughput di decine di migliaia di transazioni al secondo. Senza di esso, la latenza di propagazione ed i limiti di banda sarebbero dei problemi impossibili da risolvere per la scalabilità di Solana.

3.4 Gulf Stream

Il modulo Gulf Stream di Solana sostituisce la classica mempool con un sistema dinamico di forwarding delle transazioni verso i validatori designati per i futuri slot. Questo approccio permette di ridurre drasticamente la latenza di ordinamento e di alleggerire il carico di memoria sui nodi. In questo capitolo, approfondirò il funzionamento tecnico di Gulf Stream, il suo impatto sulla pipeline di esecuzione delle transazioni e la resilienza che conferisce all'intera rete.

3.4.1 Funzionamento di forwarding anticipato

- **Predizione del leader:** grazie alla sequenza PoH, ogni nodo validatore nella rete è in grado di conoscere in anticipo l'identità del validatore che assumerà il ruolo di leader per i prossimi slot di produzione di blocchi. Non è necessaria una comunicazione aggiuntiva per identificare il prossimo produttore di blocchi.
- **Invio diretto delle transazioni:** i client inoltrano le transazioni direttamente al validatore futuro, firmandole con un recente hash PoH come nonce temporale.

- **Validazione preliminare:** il validatore ricevente verifica immediatamente la validità delle firme e dei requisiti di account, preparando le transazioni prima ancora di proporle in un blocco.

3.4.2 Eliminazione della mempool e cache locale

- **Nessuna mempool pubblica:** non esiste un'area di attesa globale e condivisa per le transazioni in sospeso. Questo elimina il rischio di una mempool congestionata, che può portare a tempi di attesa prolungati per le transazioni ed a costi elevati in periodi di alta domanda. Ogni validatore mantiene solo una "cache locale" delle transazioni che si aspetta di processare per il proprio slot di leadership e per gli slot futuri vicini.

- **Caching distribuito:** le transazioni circolano attraverso un grafo di caching che permette ai validatori vicini di condividere rapidamente liste di transazioni non ancora incluse. Questo assicura che una transazione non dipenda da un singolo punto di fallimento per la sua inclusione e che possa essere re-inoltrata se il leader previsto non riesce a includerla.

- **Pulizia automatica:** al termine di uno slot, le transazioni non elaborate vengono automaticamente scartate dalla cache locale del validatore o re-inoltrate ai leader successivi previsti. Questo meccanismo di "garbage collection" automatica previene l'accumulo di transazioni obsolete e mantiene la memoria dei nodi ottimizzata.

3.4.3 Vantaggi prestazionali

- **Riduzione della latenza:** le transazioni possono essere eseguite non appena inizia lo slot del leader designato, anziché attendere la conferma dell'inclusione in mempool pubblica. Questo porta a tempi di consenso quasi istantanei delle transazioni dal punto di vista dell'utente, migliorando l'esperienza generale.

- **Maggiore throughput:** il caricamento e la validazione anticipata delle transazioni, consentono ai validatori di sfruttare al meglio le proprie risorse di calcolo e di rete. Al posto di dover validare e processare le transazioni in tempo reale al momento della costruzione del blocco, gran parte del lavoro è già stata fatto, incrementando il numero totale di transazioni al secondo.

- **Minore contesa di risorse:** poiché ogni validatore riceve solo il proprio sottoinsieme di transazioni, si riducono i conflitti di accesso e gli overhead di sincronizzazione che si verificano nelle mempool condivise. Questo permette una gestione più efficiente delle risorse di sistema di ciascun nodo.

3.4.4 Tolleranza ai guasti e continuità della rete

- **Resilienza al fallimento del leader:** se un validatore leader non è raggiungibile o fallisce nel produrre un blocco, le transazioni vengono automaticamente reindirizzate ed inoltrate al successivo leader previsto nella sequenza PoH. Questo garantisce che le transazioni non vadano perse e che la rete possa continuare ad andare avanti anche in caso di interruzioni impreviste dei validatori.

- **Mitigazione del congestionamento:** in caso di picchi transazionali o di elevata domanda di rete, Gulf Stream bilancia dinamicamente il carico distribuendo le transazioni su più validatori futuri. Questo previene la formazione di colli di bottiglia e di "mempool" che si gonfiano e rallentano l'intera rete, come avviene in altre blockchain.

- **Recupero da partizioni di rete:** le cache locali mantengono le transazioni in attesa fino al ripristino della connettività, in caso di partizioni temporanee della rete. Ciò assicura che nessun client perda le proprie operazioni anche in condizioni di rete meno ideali, garantendo una maggiore affidabilità.

3.5 Sealevel

Sealevel è il motore di esecuzione parallela degli smart contract di Solana, progettato per sfruttare appieno l'hardware moderno e permettere un throughput elevato. In questo capitolo, parlerò in dettaglio della sua architettura rivoluzionaria, dei modelli di concorrenza adottati e delle implicazioni profonde che ha per la scalabilità dell'intera rete Solana.

3.5.1 Motivazioni ed architettura

L'esecuzione degli smart contract tradizionalmente segue un modello sequenziale, che limita drasticamente il throughput. Sealevel rompe questo paradigma introducendo un modello completamente parallelo:

- **Parallellismo basato sugli account:** gli smart contract su Solana dichiarano in anticipo tutti gli account che andranno a leggere o modificare. Questo consente al runtime Sealevel di schedulare in parallelo tutte le istruzioni che non si sovrappongono negli account.
- **Approccio "Compute Budgeting":** ogni transazione specifica un "budget" computazionale, evitando operazioni che potrebbero causare congestione e bloccaggio dell'interprete.
- **Indipendenza dallo stato globale:** le istruzioni vengono isolate in base agli account coinvolti, riducendo i conflitti e migliorando l'efficienza.

Questa architettura permette l'esecuzione simultanea di migliaia di smart contract in una singola unità di tempo, abilitando una scalabilità orizzontale reale.

3.5.2 Modello di concorrenza

Sealevel implementa un sistema di concorrenza deterministica che si basa su due concetti fondamentali: il controllo degli accessi agli account e la gestione predittiva delle dipendenze.

Analisi statica delle dipendenze

Quando una transazione viene proposta, viene prima analizzata per determinare:

- Gli account che verranno letti (read set).
- Gli account che verranno scritti (write set).

Sealevel costruisce un grafo delle dipendenze tra transazioni e identifica i conflitti di accesso, consentendo l'esecuzione parallela solo di quelle che operano su insiemi disgiunti.

Parallel Scheduler ottimizzato

Il runtime Sealevel implementa uno scheduler parallelo che sfrutta al massimo i core CPU disponibili:

- Le istruzioni vengono eseguite in thread multipli, finché non emergono conflitti nel write set.
- L'accesso concorrente in lettura è permesso, ma ogni conflitto di scrittura viene gestito serializzando l'esecuzione delle istruzioni coinvolte.

3.5.3 Compatibilità con il modello Solana

L'integrazione di Sealevel nel sistema Solana è resa possibile dal particolare modello di "Account-based architecture" e dalla programmazione deterministica degli smart contract:

- **Programmazione via BPF (Berkeley Packet Filter):** gli smart contract scritti per Sealevel sono compilati in BPF, un linguaggio di basso livello, sicuro ed ad alte prestazioni che può essere eseguito in ambienti isolati (Sandboxed) all'interno del kernel del sistema operativo. Questa scelta garantisce un'esecuzione efficiente e sicura, poiché il codice BPF ha accesso limitato e controllato alle risorse di sistema.

- **Chiamate incrociate asincrone:** Sealevel consente a un programma di invocare altri programmi in maniera asincrona, permettendo di interagire tra loro in modo efficiente pur mantenendo la sicurezza dell'esecuzione.

- **Isolamento della memoria:** l'esecuzione dei contratti è Sandboxed: ogni programma ha accesso solo agli account ed alla memoria specificatamente dichiarati all'inizio della transazione. Questo isolamento impedisce che un programma malevolo possa accedere o modificare dati a cui non è autorizzato, migliorando drasticamente la sicurezza dell'intera rete e proteggendo gli account degli utenti.

3.5.4 Vantaggi

- **Scalabilità orizzontale:** l'esecuzione parallela delle transazioni permette di scalare il throughput della rete quasi linearmente con l'aumento del numero di core CPU e GPU disponibili sui validatori. Man mano che l'hardware diventa più potente, anche la capacità di elaborazione di Solana aumenta proporzionalmente.

- **Riduzione della latenza:** le transazioni indipendenti vengono eseguite simultaneamente anziché sequenzialmente, riducendo drasticamente il tempo medio di conferma per la maggior parte delle transazioni.

- **Efficienza computazionale:** il modello di "Compute Budgeting" evita l'utilizzo eccessivo delle risorse computazionali. Le transazioni vengono interrotte se superano il loro budget, garantendo che l'hardware sia utilizzato in modo efficiente e che la rete non venga rallentata da calcoli inutili o malevoli.

3.5.5 Limiti

- **Conflitti di scrittura:** le transazioni che modificano lo stesso account (cioè, hanno un write set che si sovrappone) devono necessariamente essere eseguite in modo seriale. Questo è un requisito fondamentale per mantenere la coerenza dello stato. Sebbene Sealevel massimizzi il parallelismo dove possibile, i conflitti di scrittura possono ancora creare colli di bottiglia in applicazioni ad alta contesa.

- **Programmazione più complessa:** gli sviluppatori devono dichiarare esplicitamente tutti gli account che un programma intende leggere o scrivere in anticipo, il che rende la logica di programmazione più rigida e meno flessibile rispetto a modelli in cui gli accessi agli account possono essere determinati dinamicamente. Questo può aumentare la complessità dello sviluppo di smart contract su Solana.

- **Debugging e Testing:** la natura parallela di Sealevel introduce sfide aggiuntive per il debugging e testing predittivo di smart contract complessi. Gli sviluppatori devono adottare metodologie di test più sofisticate per garantire la robustezza dei loro programmi.

3.6 Pipelining

Il pipelining in Solana è un meccanismo che sovrappone e parallelizza le diverse fasi del processamento delle transazioni, riducendo drasticamente la latenza complessiva e massimizzando l'utilizzo delle risorse hardware. In questo capitolo, approfondisco il funzionamento del pipelining, i suoi componenti principali, i benefici in termini di throughput e i possibili limiti.

3.6.1 Funzionamento delle pipeline interne

Solana suddivide l'elaborazione di blocchi e transazioni in una serie di stage distinti, ciascuno dei quali può operare in parallelo su batch differenti di dati:

1. **Ricezione dei pacchetti (fetch)**: i nodi validatori ricevono i pacchetti UDP propagati da Turbine e ne eseguono un primo filtraggio (checksum, autenticità).

2. **Verifica delle firme (signature verification)**: tramite batch su GPU o CPU, le firme Ed25519 vengono verificate in parallelo per centinaia di transazioni contemporaneamente.

3. **Scheduling e analisi delle dipendenze (banking stage)**: il runtime esamina le transazioni per identificare conflitti di accesso allo stato (account read/write), raggruppando quelle non conflittuali.

4. **Esecuzione degli smart contract (execution)**: sfruttando Sealevel, ogni gruppo di transazioni indipendenti viene eseguito simultaneamente su core CPU e, quando possibile, su istruzioni SIMD delle GPU.

5. **Aggiornamento dello stato (write back)**: le modifiche agli account vengono scritte sul database Cloudbreak in modalità Memory-mapped, con operazioni I/O parallele su SSD.

6. **Broadcast dei risultati (broadcast)**: i risultati finali (nuovo stato e firme di conferma) vengono propagati ai peer per completare il consenso Tower BFT.

Ogni stage lavora su un batch indipendente: mentre uno stage elabora il batch n, il successivo può già iniziare a processare il batch $n-1$, creando una catena di produzione continua che sfrutta al massimo la pipeline.

3.6.2 Vantaggi

- **Riduzione della latenza:** il pipelining elimina i tempi morti tra le diverse fasi di elaborazione, permettendo un tempo di blocco (il tempo tra la produzione di due blocchi consecutivi) inferiore al secondo, anche in condizioni di rete non ideali.

- **Parallelismo a più livelli:** ogni fase della pipeline gira in parallelo su thread e core separati, garantendo un elevato throughput sostenuto. Questo approccio multi-livello al parallelismo (dalla verifica delle firme, all'esecuzione degli smart contract) è ciò che consente a Solana di gestire un volume di transazioni significativamente più elevato rispetto ad altre blockchain.

- **Basso overhead di contesa:** la suddivisione in batch e la pre-analisi delle dipendenze (grazie a Sealevel) riducono drasticamente i conflitti di accesso allo stato, migliorando la scalabilità orizzontale. Questo minimizza i rallentamenti dovuti a "lock" o "mutex" e migliora la scalabilità orizzontale, permettendo al sistema di crescere con l'aggiunta di più risorse computazionali.

- **Scalabilità elastica:** aggiungendo componenti hardware (core CPU, GPU, SSD), la pipeline scala automaticamente. Non è necessaria una

riprogettazione complessa dell'architettura per supportare un aumento del carico transazionale, rendendo Solana capace di adattarsi alle crescenti esigenze della rete.

3.6.3 Limiti

- **Bilanciamento dei stage:** l'efficienza complessiva di una pipeline è determinata dal suo stadio più lento. Se uno stage diventa troppo lento (ad esempio, a causa di operazioni I/O su SSD saturi o di un eccessivo carico computazionale su un singolo core), può ridurre l'efficienza complessiva dell'intera pipeline, rallentando il flusso di transazioni. È fondamentale un bilanciamento costante delle risorse e capacità di ogni stadio.

- **Gestione degli errori:** se un errore si verifica in uno stadio avanzato, per un particolare batch di transazioni, il rollback ed il retry di quel singolo batch devono essere gestiti con attenzione per evitare inconsistenze di stato o di propagazione, specialmente in un ambiente decentralizzato.

- **Complessità implementativa:** l'organizzazione delle pipeline, la sincronizzazione delle code tra i diversi stage e la gestione degli stati intermedi introduce una notevole complessità nel codice del validatore. Questo richiede una progettazione software robusta e una manutenzione attenta per garantire stabilità e prestazioni.

3.6.4 Contributo alla Scalabilità di Rete

Il Pipelining rappresenta uno degli elementi principali per il modello "L1-native scaling" adottato da Solana:

- Consente di mantenere un elevato utilizzo delle risorse computazionali per ogni core disponibile. Massimizzando il lavoro svolto da ogni unità di elaborazione, Solana può sfruttare al meglio le prestazioni offerte dall'hardware moderno.

- Questo meccanismo riduce drasticamente i tempi di elaborazione "end-to-end" di ogni singola transazione. Ciò permette a Solana di avvicinarsi a cifre di 5.000–10.000 transazioni al secondo (TPS) effettive nella sua mainnet-beta e di scalare fino a decine di migliaia di TPS su infrastrutture ottimizzate, posizionandosi come una delle blockchain più veloci.

- Si integra perfettamente con altri moduli chiave come PoH, Gulf Stream e Cloudbreak. Questo crea un flusso continuo e armonizzato di dati e transazioni che scorre attraverso la rete senza strozzature significative, ottimizzando ogni fase del ciclo di vita di una transazione e garantendo che tutti i componenti lavorino insieme per massimizzare il throughput.

3.7 Cloudbreak (Accounts database)

Cloudbreak é un database progettato per gestire milioni di account in maniera veloce e scalabile. Il compito principale di Cloudbreak é tenere traccia dello stato degli account. Ogni volta che viene eseguita una transazione, i profili coinvolti vengono letti e aggiornati nel minor tempo possibile. In questo capitolo, entreró in dettaglio sul funzionamento di Cloudbreak, sulla pulizia e gestione dello spazio e sulla sua scalabilità, consistenza e resilienza.

3.7.1 Funzionamento di Cloudbreak

Per comprendere meglio Cloudbreak, é utile analizzare il suo funzionamento durante l'esecuzione di un blocco. Ogni validatore lo utilizza per leggere lo stato degli account coinvolti in una transazione, aggiornare tali dati e garantire che il risultato sia consistente e persistente nel tempo, applicando i seguenti passaggi:

1. **Caricamento dello stato iniziale**: all'inizio dell'elaborazione di un blocco, il validatore accede ad uno snapshot dello stato corrente tramite memory-mapped files, in modo da avere una base di riferimento immutabile per il passo successivo.

2. **Esecuzione parallela delle transazioni**: grazie al runtime Sealevel, più transazioni vengono eseguite in parallelo. Cloudbreak fornisce un accesso concorrente lock-free alla memoria, permettendo a ciascun thread di leggere i dati necessari senza interferenze.

3. **Aggiornamento dello stato**: se una transazione modifica lo stato di un account, viene creato un nuovo record per l'account aggiornato, cosí che le altre transazioni in esecuzione continuino a leggere la versione originale.

4. **Batching delle scritture**: al termine del blocco, le modifiche vengono raccolte in un batch e scritte in modo atomico sui dischi SSD.

3.7.2 Pulizia e gestione dello spazio

Cloudbreak include una serie di meccanismi per la pulizia e gestione dello spazio di archiviazione per evitare che il database diventi ingombrante e lento:

- **Archiviare lo storico**: Cloudbreak, nonostrante gestisca lo stato corrente, non è responsabile dello storage a lungo termine dell'intera storia delle transazioni. Questo compito è delegato agli Archivers, che scaricano i dati storici e li conservano in modo decentralizzato. Questa separazione dei compiti alleggerisce il carico sui validatori e sul database Cloudbreak, permettendo a quest'ultimo di concentrarsi sulla gestione dello stato più recente.

- **Garbage Collection selettiva**: Cloudbreak implementa meccanismi di "garbage collection" per rimuovere account o dati obsoleti che non sono più necessari per il funzionamento dello stato corrente. Questi processi sono progettati per essere efficienti e per non interferire con le operazioni di transazione ad alta velocità.

- **Stato "Compact"**: l'obiettivo è mantenere lo stato di Cloudbreak il più compatto possibile. Questo include l'ottimizzazione delle strutture dati e degli indici per minimizzare l'ingombro su disco e massimizzare la velocità di accesso.

3.7.3 Scalabilità orizzontale, Consistenza e Resilienza di Cloudbreak

Oltre al design di base per l'accesso concorrente allo stato, é importante comprendere come Cloudbreak assicuri la scalabilità orizzontale, mantenga la consistenza dei dati e tolleri guasti hardware o di rete:

- **Striping RAID 0 e mappatura in memoria:** Cloudbreak organizza i dati degli account su più SSD in striping RAID 0, mappando ciascun file su memoria virtuale tramite mmap. Questo approccio permette di distribuire le richieste di I/O tra tutti i dischi disponibili, riducendo drasticamente la latenza di lettura e scrittura ed aumentando linearmente il throughput con l'aggiunta di ogni nuovo SSD.

- **Accesso concorrente e lock-free reads:** grazie alla mappatura in memoria, le operazioni di lettura possono essere eseguite in modalità lock-free: ogni thread legge direttamente dal proprio segmento di memoria mappata senza competere per mutex o lock a livello di file.

Le operazioni di scrittura vengono coordinate tramite buffer di batching che raccolgono più aggiornamenti e li scrivono periodicamente sui dischi, minimizzando la contesa e l'overhead di sincronizzazione tra thread.

- **Consistency model:** all'inizio di ogni blocco, il validatore fissa un "bookmark" sullo stato corrente (snapshot), garantendo che tutte le transazioni incluse in quel blocco leggano uno stato immutabile. Le modifiche vengono applicate in modo atomico solo al termine dell'elaborazione del blocco, evitando letture incoerenti o "dirty reads" durante l'esecuzione parallela su Sealevel.

- **Tolleranza ai guasti dei dischi:** pur non includendo ridondanza hardware completa, Cloudbreak sfrutta il fatto che ogni account é replicato su più validatori: in caso di guasto di un SSD, i dati possono essere recuperati leggendo il medesimo snapshot dallo storage di un altro validatore. Inoltre, i validatori possono "ricostruire" lo stato mancato eseguendo nuovamente le transazioni dal ledger a partire dall'ultimo snapshot sano.

- **Recupero incrementale e Compaction:** per evitare che i file mappati crescano indefinitamente, Cloudbreak effettua periodicamente operazioni di "Compaction", dove raccoglie i delta di stato in un nuovo file snapshot, rilasciando lo spazio su disco occupato dalle vecchie versioni. Questo processo é eseguito in background ed assicura che lo storage rimanga efficiente anche dopo migliaia di blocchi.

3.8 Archivers

Gli Archivers sono nodi specializzati incaricati di conservare la storia completa del ledger, alleggerendo i validatori dal carico associato alla memorizzazione persistente dei dati. In questo capitolo, approfondirò le componenti crittografiche e di sistema che governano il loro funzionamento, illustrando i meccanismi di frammentazione, verifica e resilienza.

3.8.1 Funzionamento: Erasure Coding e Proofs of Replication

- **Erasure Coding:** quando un blocco di dati storici viene scaricato dal leader e passato agli Archivers, viene applicato, su di esso, una tecnica

di "Erasure Coding" (codifica a cancellazione), come i codici Reed-Solomon. Questa tecnica frammenta i dati in più parti e genera dati di parità ridondanti, in modo che il file originale possa essere ricostruito, anche se una certa percentuale di queste parti dovesse essere persa o danneggiata. Questo garantisce un'elevata disponibilità e robustezza dei dati storici archiviati, anche se alcuni Archivers dovessero diventare inattivi o malevoli.

- **Proofs of Replication (PoRep):** per garantire che gli Archivers stiano effettivamente immagazzinando i dati in modo corretto e completo, Solana utilizza meccanismi di PoRep, una prova crittografica che dimostra che un Archiver ha immagazzinato correttamente un'intera porzione di dati specifici. Questo meccanismo previene attacchi come "Sybil attacks" (in cui un singolo attore malevolo simula di essere più nodi per controllare una porzione maggiore della rete) o "outsourcing attacks" (in cui un Archiver dichiara di immagazzinare dati ma in realtà li ha eliminati o non li ha mai scaricati). Gli Archivers devono produrre periodicamente queste prove verificabili per dimostrare la loro onestà e la loro continua conservazione dei dati.

3.8.2 Integrazione con PoH

- **Temporalità delle challenge:** le challenge PoRep sono derivate dall'hash di posizioni specifiche nella catena PoH, garantendo che un Archiver non possa pre-computare o riutilizzare prove.

- **Verifica distribuita:** i validatori raccolgono le PoRep inviate dagli Archivers e ne validano l'autenticità ricontrollando la catena PoH e la merkle root dei frammenti.

- **Sincronizzazione e incentivi:** solo gli Archivers che presentano PoRep corrette entro una finestra temporale stabilita (basato su PoH) ricevono ricompense in SOL, incentivando la partecipazione e la corretta conservazione dei dati.

3.8.3 Verifica periodica e Gestione delle contromisure

- **Verifica periodica (Challenges):** le challenge non seguono un calendario prevedibile, ma sono pseudo-randomizzate attraverso PoH, aumentando la difficoltà di attacco mirato. I validatori o altri nodi della rete possono inviare challenges casuali agli Archivers, chiedendo loro di

fornire Proofs of Replication per specifiche porzioni di dati che affermano di detenere. Archivers inattivi o che falliscono le PoRep subiscono slashing parziale dello stake o esclusione temporanea dalla rete, mentre quelli affidabili guadagnano premi proporzionali alle proof inviate.

- **Rotazione dei frammenti:** il sistema riassegna, periodicamente, i frammenti agli Archivers per mitigare l'accumulo di dati non utilizzati e per ridurre il rischio di compromissione mirata.
- **Gestione delle contromisure:** In caso di Archivers malevoli o non responsivi, i dati mancanti o compromessi possono essere recuperati da altri Archivers che detengono copie ridondanti (grazie all'Erasure coding). Nuovi Archivers possono essere reclutati e incentivati a replicare i dati necessari, garantendo che la disponibilità della storia del ledger non sia compromessa.

3.8.4 Tolleranza ai guasti e Disponibilità

- **Ridondanza:** Grazie agli "Erasure Codes", la rete può tollerare la perdita fino a k frammenti per blocco (dove k è scelto secondo il livello di ridondanza configurato) senza compromettere la capacità di recupero. Le copie ridondanti ed i dati di parità consentono la ricostruzione del ledger.
- **Recupero dinamico:** In caso di crash o malfunzionamento di un Archiver, gli altri nodi specializzati restituiscono rapidamente i frammenti mancanti, supportati dal meccanismo PoRep e dai challenge PoH.
- **Scalabilità orizzontale:** È possibile aggiungere nuovi Archivers "on the fly": una volta sincronizzati con l'ultimo stato PoH, ricevono i frammenti e iniziano subito a generare PoRep, estendendo linearmente la capacità di archiviazione dello storico.

3.8.5 Contributo alla scalabilità della rete

- **Senza la necessità di archiviare l'intera storia, i validatori possono mantenere le loro risorse dedicate al throughput ed alla latenza delle transazioni correnti.**
- **Permette una maggiore decentralizzazione:** poiché i requisiti hardware per i validatori sono ridotti (non devono immagazzinare terabyte di dati storici), più attori possono partecipare al consenso, aumentando la decentralizzazione e la sicurezza della rete.

- **Supporta la crescita esponenziale del ledger:** Man mano che Solana processa sempre più transazioni, la storia del ledger crescerà esponenzialmente. Gli Archivers sono progettati per scalare con questa crescita, fornendo una soluzione di storage sostenibile a lungo termine.

Capitolo 4

Algoritmi crittografici principali

In questo capitolo, vengono analizzati gli algoritmi crittografici principali che supportano il funzionamento e la sicurezza di Solana. In particolare, viene esaminata la pipeline di verifica transazionale, che sfrutta Ed25519 per le firme digitali e SHA-256 per il Proof of History, e si approfondisce il meccanismo di Proof of Replication che, integrando tecniche basate su SeDRAM, si propone come strumento di difesa contro gli attacchi Sybil.

4.1 Pipeline di verifica transazionale: Ed25519 per firme digitali e SHA-256 per PoH

La sicurezza e velocità della rete Solana si fondano su una pipeline crittografica accuratamente progettata, in cui due algoritmi giocano un ruolo fondamentale:

4.1.1 Ed25519 per firme digitali

- **Efficienza e velocità:** Ed25519 è una variante dell'algoritmo a curva ellittica ed è più veloce nella generazione e verifica delle firme rispetto ad altri algoritmi di firma digitale ampiamente utilizzati (come RSA o ECDSA in alcune configurazioni). Questa rapidità è fondamentale per Solana, che punta a migliaia di transazioni al secondo. I validatori possono verificare in parallelo centinaia di firme Ed25519 in pochi millisecondi, spesso sfruttando le capacità di calcolo parallelo delle GPU.

- **Sicurezza:** l'algoritmo è costruito su basi matematiche solide ed è resistente ad una vasta gamma di attacchi crittografici noti. La sua sicurezza si basa sulla difficoltà del problema della curva ellittica discreta sulla curva 25519.

- **Non ripudiabilità e integrità:** l'impiego di Ed25519 assicura che ogni messaggio e transazione siano firmati in modo univoco, impedendo a un partecipante di negare in seguito la propria partecipazione e garantendo l'integrità dei dati trasmessi.

- **Dimensioni ridotte delle firme:** sono relativamente piccole (64 byte), il che contribuisce a ridurre il "peso" delle transazioni e dei blocchi, ottimizzando l'uso della banda di rete e la capacità di archiviazione.

4.1.2 SHA-256 per il PoH

- **Costruzione di una sequenza temporale:** SHA-256 viene eseguito in modalità iterativa per generare una catena di hash che costituisce la base temporale di PoH. Ogni output della funzione hash diventa input per il successivo ciclo, creando una sequenza ininterrotta e verificabile. Questa sequenzialità garantisce che il tempo sia "inciso" crittograficamente nella sequenza: per avanzare di un passo nel tempo, è necessario completare il calcolo della funzione hash precedente.

- **Resistenza alle collisioni:** la robustezza di SHA-256, nota per la sua bassa probabilità di collisione, garantisce che ogni step della sequenza sia univoco e resistano ad eventuali tentativi di manipolazione. Questo rende l'ordine degli eventi irreversibile e verificabile in maniera autonoma dai validatori.

- **Verificabilità rapida:** nonostante la generazione della sequenza PoH sia un processo sequenziale e computazionalmente intensivo (VDF), la verifica della sequenza è rapida. Così, qualsiasi nodo può ricalcolare una porzione della sequenza PoH per verificare la sua autenticità e l'ordine degli eventi senza dover ricalcolare l'intera storia. Questo permette ai validatori di accertare rapidamente la correttezza del PoH generato dal leader.

4.2 Proof of Replication (PoRep): SeDRAM e Resistenza agli attacchi Sybil

Il meccanismo di Proof of Replication (PoRep) rappresenta un ulteriore livello di sicurezza e verifica, che punta a dimostrare che i dati sono stati replicati in maniera adeguata nei nodi della rete. In questo contesto, il supporto tecnologico fornito da SeDRAM, unitamente a strategie di mitigazione contro gli attacchi Sybil, gioca un ruolo cruciale.

4.2.1 SeDRAM ed Ottimizzazione della replicazione

- **Memoria ad alte prestazioni:** SeDRAM (Synchronous Dynamic Random Access Memory) viene impiegata per ottimizzare il processo di replicazione dei dati, garantendo tempi di accesso ridotti e una gestione efficiente della memoria. Questo è fondamentale per validare rapidamente la presenza e l'integrità delle repliche.

- **Generazione efficiente delle prove:** essendo che i PoRep possono essere computazionalmente intensivi, Solana mira a ottimizzare il processo di generazione delle prove per gli Archivers, in modo che possano produrle regolarmente senza un consumo eccessivo di risorse, pur mantenendo un alto livello di sicurezza.

- **Compressione e frammentazione:** prima dell'archiviazione e della replicazione, i dati storici possono essere compressi e frammentati in unità gestibili. L'Erasure coding (come Reed-Solomon) viene applicato a questi frammenti per creare ridondanza, garantendo che i dati possano essere ricostruiti anche se una porzione degli Archivers fallisce o si comporta malevolmente.

- **Verifica dell'integrità dei dati:** grazie a SeDRAM, il sistema è in grado di effettuare operazioni di verifica ripetute ed ad alta velocità, assicurando che ogni replica mantenga la stessa struttura ed integrità del dato originale.

- **Auditability:** gli Archivers devono essere periodicamente "auditati" (verificati) dalla rete. Questo avviene attraverso "challenge" crittografiche che richiedono agli Archivers di dimostrare che possiedono realmente quei dati attraverso la generazione di PoRep. Queste sfide sono leggere per il verificatore e costose per l'Archiver solo se non detiene i dati.

4.2.2 Resistenza agli attacchi Sybil

- **Definizione degli attacchi Sybil:** gli attacchi Sybil si verificano quando un singolo utente crea numerose identità false per ottenere un controllo sproporzionato sulla rete. Questo tipo di attacco può compromettere la sicurezza del sistema, se non adeguatamente contrastato.

- **Mitigazione tramite PoRep:** il processo di Proof of Replication, unito alla gestione efficiente della memoria tramite SeDRAM, impone un elevato costo computazionale e di risorse per la creazione di repliche false. Quindi, replicare e dimostrare la presenza dei dati in maniera fraudolenta richiede investimenti significativi, rendendo l'attacco meno vantaggioso dal punto di vista economico.

- **Vincolo di spazio fisico:** un PoRep richiede che l'Archiver dimostri di possedere i dati e di averli replicati in modo univoco nello spazio fisico. Questo significa che l'Archiver non può semplicemente affermare di archiviare una copia di dati per ogni identità Sybil che ha creato. Deve invece dimostrare di aver effettivamente utilizzato lo spazio di archiviazione corrispondente per ogni replica. Ciò rende computazionalmente e fisicamente costoso per un attaccante Sybil simulare la detenzione di più copie di dati di quante ne abbia realmente.

- **Resistenza all'outsourcing:** un altro tipo di attacco che i PoRep sono in grado di contrastare è l'Outsourcing attack, in cui un Archiver promette di memorizzare i dati ma, in realtà, li delega ad un altro Archiver (o li elimina) e poi recupera solo le parti necessarie per rispondere alle sfide. I PoRep sono progettati per richiedere una "prova di conoscenza" dei dati originali combinata con la prova di una replica unica, rendendo difficile l'outsourcing senza essere scoperti.

- **Integrazione con Proof of Stake:** il sistema di Archivers può anche beneficiare dell'integrazione con un meccanismo di stake, dove essi devono bloccare una certa quantità di token come garanzia della loro onestà. Se falliscono un PoRep o si comportano malevolmente, il loro stake può essere soggetto a Slashing, fornendo un incentivo economico all'onestà e scoraggiante per effettuare attacchi Sybil.

Capitolo 5

Gestione chiavi e Sicurezza della rete

La sicurezza di Solana dipende sia dagli algoritmi crittografici sottostanti, sia dalla gestione efficace delle chiavi e dalla difesa contro attacchi di rete. In questo capitolo, esplorerò le soluzioni adottate in termini di architettura delle chiavi, firme crittografiche collaborative (Threshold Signatures), e tecniche avanzate per la rotazione sicura delle chiavi e la protezione contro attacchi di tipo DDoS.

5.1 Chiavi gerarchiche

Solana adotta un'architettura di chiavi pubbliche "gerarchiche" per facilitare la gestione della sicurezza, soprattutto in applicazioni ad alto volume. Il modello è ispirato a BIP32 (Bitcoin Improvement Proposal), ma adattato all'ecosistema Solana. Le caratteristiche principali includono:

- **Derivazione deterministica:** le chiavi derivate da una chiave master possono essere rigenerate in qualsiasi momento, purché si conosca la chiave iniziale e il percorso di derivazione, migliorando la portabilità e la sicurezza.
- **Isolamento tra livelli:** ogni livello della gerarchia può essere isolato, limitando l'impatto di una potenziale compromissione su sottoinsiemi specifici di chiavi.
- **Gestione di accessi granulari:** tramite percorsi di derivazione specifici, è possibile creare sotto-chiavi con permessi limitati, ideali per deleghe o sessioni temporanee.

5.2 Threshold Signatures

Per migliorare la robustezza contro attacchi e guasti, Solana può impiegare "Threshold Signatures" basate su schemi crittografici come Shamir's Secret Sharing o protocolli compatibili con Ed25519, così da permettere ad un insieme di validatori o nodi fidati di firmare transazioni in modo distribuito.

- **Collaborazione sicura:** una firma é valida solo quando almeno t su n partecipanti collaborano, aumentando la resilienza a compromissioni parziali.
- **Assenza di single point of failure:** nessun partecipante possiede l'intera chiave privata, riducendo il rischio di compromissione totale.
- **Riduzione del carico computazionale:** alcune implementazioni permettono aggregazione e compressione delle firme, migliorando l'efficienza delle transazioni su larga scala.

5.3 Rotazione delle chiavi

Solana supporta rotazioni sia automatiche (via smart contract) sia manuali per i wallet, account e validatori. Le motivazioni principali includono:

- **Prevenzione da compromissioni prolungate:** una chiave che potrebbe essere stata acquisita da un attaccante, ma che non è ancora stata sfruttata, viene efficacemente neutralizzata e resa obsoleta non appena viene sostituita da una nuova chiave. Questo riduce la "finestra di esposizione" di una chiave e limita il tempo durante il quale un attaccante potrebbe agire indisturbato. La rotazione, quindi, non è solo una reazione ad un incidente, ma una strategia di mitigazione del rischio.
- **Verificabilità ed accountability:** i meccanismi di aggiornamento e rotazione delle chiavi sono progettati per essere registrati on-chain, rendendo possibile le veriche formali. Questa caratteristica è di vitale importanza per scopi di verifica e per garantire la trasparenza. Le organizzazioni e gli utenti possono formalmente verificare quando e come le chiavi sono state modificate, chi ha autorizzato tali modifiche e se le procedure di sicurezza interne sono state rispettate. Questo contribuisce a creare un ecosistema più trasparente ed affidabile.

5.4 Mitigazione DDoS con Rate-Limiting crittografico

Essendo che gli attacchi DDoS (Distributed Denial of Service) rappresentano una minaccia significativa per la blockchain, Solana adotta un sistema di "rate-limiting crittografico", utile per mantenere un elevato throughput anche in presenza di tentativi malevoli di congestione, che opera sui seguenti principi:

- **Proof-of-Stake come filtro economico:** ogni nodo che invia transazioni deve dimostrare il possesso di stake. Questo serve come barriera economica per ridurre spam e flood.
- **Verifica crittografica del volume:** il sistema può associare, ad ogni account, limiti transazionali determinati dal volume di firma e dallo stake sottostante. Se un attore supera tale soglia, viene temporaneamente penalizzato o isolato.
- **Hardware-enforced quotas:** alcuni validatori possono adottare rate limiter a livello di rete per bloccare in anticipo pacchetti malformati o duplicati.

Capitolo 6

Crittografia applicata: smart contract e NFT

In questo capitolo, analizzo le applicazioni della crittografia all'interno del sistema Solana, con particolare riferimento a smart contract e NFT: l'uso delle Verifiable Delay Functions (VDF) come elemento computazionale nei contratti intelligenti, lo standard Solana Program Library (SPL) Token con supporto per metadati cifrati, e l'integrazione progressiva dei ZK-SNARKs come strumento per abilitare la privacy on-chain.

6.1 Sfida VDF nei Contratti Intelligenti su Solana

Le VDF sono funzioni computazionalmente intensive, progettate per richiedere un determinato tempo sequenziale per essere calcolate, ma facilmente verificabili. Su Solana, l'integrazione delle VDF è considerata un possibile strumento per:

- **Incentivare on-chain randomness verificabile:** le VDF sono utilizzate come meccanismo per generare numeri casuali resistenti alla manipolazione.
- **Limitare front-running:** l'introduzione di un ritardo computazionale obbligatorio mitiga i rischi derivanti dall'osservazione delle transazioni mempool e successive manipolazioni da parte di validatori opportunistici.

- **Proof-of-Time on-chain:** le VDF possono rappresentare un'integrazione interessante con PoH per task contrattuali che richiedano latenza verificabile indipendente.

Le VDF su Solana possono essere implementate via smart contract oppure attraverso moduli off-chain collegati via syscall e verificati attraverso SHA-256 o gruppi RSA modulari.

6.2 Standard SPL Token e Metadati cifrati

Lo standard SPL definisce le specifiche per token fungibili e non fungibili. Oltre alla semplice rappresentazione di bilanci e identificatori, lo standard supporta estensioni crittografiche, tra cui i metadati cifrati.

6.2.1 Metadati cifrati

- **Crittografia simmetrica AES-GCM:** per la cifratura dei metadati NFT, gli sviluppatori possono utilizzare lo standard di crittografia simmetrica AES in modalità Galois/Counter Mode (GCM). AES-GCM è una scelta robusta e ampiamente adottata per la sua efficienza e sicurezza. Questa modalità assicura la confidenzialità dei dati (rendendoli illeggibili senza la chiave corretta) e garantisce anche la loro integrità ed autenticità tramite un "tag di autenticazione". Ciò significa che qualsiasi tentativo di manomissione dei metadati cifrati verrebbe immediatamente rilevato. Prima di essere caricati on-chain o su sistemi di storage decentralizzati, i metadati sensibili possono essere cifrati, proteggendone la privacy.

- **Gestione chiavi con derived addresses:** l'accesso ai contenuti cifrati può essere gestito in modo intelligente e sicuro attraverso l'uso di "Program-derived addresses" (PDAs). Sono indirizzi che non hanno una chiave privata associata nel senso tradizionale, ma la cui proprietà è derivata da un programma (smart contract) e da un insieme di semi. Questo permette alle PDAs di fungere da proxy crittografici autorizzati. Possono essere configurate per detenere o generare le chiavi simmetriche necessarie alla decifratura dei metadati, consentendo solo a programmi specifici o a logiche predefinite di accedere alle informazioni riservate.

Questa struttura consente un livello superiore di privacy e controllo, utile in applicazioni come musica, arte, giochi, certificati o documentazione legale su blockchain.

6.3 ZK-SNARKs per transazioni private

Sebbene Solana non abbia inserito nativamente un supporto full-stack per ZK-SNARKs (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge), si stanno sviluppando librerie e protocolli compatibili per l'esecuzione di "circuiti zk" all'interno di smart contract personalizzati. Le caratteristiche principali sono:

- **Privacy su conti e trasferimenti:** l'introduzione di ZK-SNARKs può permettere il trasferimento di token o NFT senza rivelare mittente, destinatario o quantità, garantendo al contempo la validità della transazione.
- **ZK-Rollup compatibili:** layer costruiti sopra Solana possono delegare la verifica di centinaia di transazioni off-chain ad un'unica prova SNARK verificata on-chain.
- **Ottimizzazione computazionale:** grazie alla struttura ad alta parallelizzazione di Solana, è possibile prevedere l'accelerazione della verifica SNARK tramite GPU, permettendo l'uso in tempo reale di microtransazioni private.

Bibliografia

[1] A. Yakovenko, *Solana: A new architecture for a high performance blockchain*, 2018. Disponibile su: https://solana.com/solana-whitepaper.pdf

[2] Solana Documentation, 2023. Disponibile su: https://solana.com/docs

[3] Solana Foundation, *Proof of History: A Clock for Blockchain*, 2018. Disponibile su: https://solana.com/news/proof-of-history--a-clock-for-blockchain

[4] Solana Foundation, *Tower BFT: Solana's High Performance Implementation of PBFT*, 2018. Disponibile su: https://solana.com/news/tower-bft--solana-s-high-performance-implementation-of-pbft

[5] Solana Foundation, *Turbine — Solana's Block Propagation Protocol Solves the Scalability Trilemma*, 2019. Disponibile su: https://solana.com/news/turbine---solana-s-block-propagation-protocol-solves-the-scalability-trilem

[6] Solana Foundation, *Gulf Stream — Solana's Mempool-Less Transaction Forwarding Protocol*, 2019. Disponibile su: https://solana.com/news/gulf-stream--solana-s-mempool-less-transaction-forwarding-protocol

[7] Solana Foundation, *Sealevel — Parallel Processing Thousands of Smart Contracts*, 2019. Disponibile su: https://solana.com/news/sealevel---parallel-processing-thousands-of-smart-contracts

[8] Solana Foundation, *Pipelining in Solana: The Transaction Processing Unit*, 2019. Disponibile su: https://solana.com/news/pipelining-in-solana-the-transaction-processing-unit

[9] Solana Foundation, *Cloudbreak — Solana's Horizontally Scaled State Architecture*, 2019. Disponibile su: https://solana.com/news/cloudbreak---solana-s-horizontally-scaled-state-architecture

[10] Solana Foundation, *Archivers — Solana's Solution to Petabytes of Blockchain Data Storage*, 2019. Disponibile su: https://solana.com/news/archivers---solana-s-solution-to-petabytes-of-blockchain-data-storage

[11] M. Castro e B. Liskov, *Practical Byzantine Fault Tolerance*, in OSDI, 1999. Disponibile su: https://pmg.csail.mit.edu/papers/osdi99.pdf

[12] D. J. Bernstein e T. Lange, *High-speed high-security signatures*, 2012. Disponibile su: https://ed25519.cr.yp.to/

[13] National Institute of Standards and Technology (NIST), *FIPS PUB 180-4: Secure Hash Standard (SHS)*, 2015. Disponibile su: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

[14] Protocol Labs, *Proof-of-Replication: Scalable Decentralized Storage*, 2017. Disponibile su: https://filecoin.io/proof-of-replication.pdf

[15] Project Serum, *Anchor Book - Solana Smart Contract Framework*, 2022. Disponibile su: https://book.anchor-lang.com/

[16] Jump Crypto, *Firedancer: A High Performance Validator for Solana*, 2023. Disponibile su: https://jumpcrypto.com/firedancer/

[17] Protocol Labs, *The Future of Zero Knowledge Proofs*, 2023. Disponibile su: https://www.protocol.ai/protocol-labs-the-future-of-zk-proofs.pdf

[18] D. Boneh et al., *ZK-SNARKs: Scalable, Transparent, and Post-Quantum Secure Proofs*, 2018. Disponibile su: https://eprint.iacr.org/2018/046.pdf