GT 50 改善
VISIONARY INNOVATORS

NOTARIZZAZIONE SCALABILE ED AUTENTICATA DI DATI IOT

Report 2:

studio di metodi innovativi sharding-basedper estendere la notarizzazione di grandi moli di dati prodotti da dispositivi IoT all'uso di piattaforme evolute di smart-contract.

Con la collaborazione di:
Università di Trento

UNIVERSITÀ DI TRENTO
Dipartimento di Matematica
Laboratorio di Matematica Industriale e Crittografia

Fornitore:
GT50 Srl

GT50 改善
VISIONARY INNOVATORS

Attività svolta nell'ambito dell'Avviso promosso dal Ministero dell'Università e della Ricerca per la presentazione di Idee progettuali per Smart Cities and Communities and Social Innovation di cui al D.D. n. 391/Ric. del 5 luglio 2012 e ss.mm.ii. - SIN_00968 THE LEARNING METERS NETWORK: workpackage formativo del SCN_00398 - CUP J49G14000140008.

GT 50 改善
VISIONARY INNOVATORS

Prj Università di Perugia
Dipartimento Ingegneria

Ordine 5927252 - CIG ZE32F314CA

Consenso BFT su Blockchain con Sharding

Abstract

Nel presente documento presentiamo uno studio sulla possibilità di implementare un algoritmo di consenso che sia BFT su blockchain di nuova generazione, nativamente suddivise in shard indipendenti.

Vengono analizzate le necessità di base di una simile blockchain e vengono individuati meccanismi e strutture atti a soddisfare queste esigenze. In particolare si vuole che i diversi shard possano riconciliare le loro transazioni e comunicare tra loro.

Per soddisfare queste due necessità introduciamo due ulteriori catene parallele, rispettivamente la Master-Chain e la Time-Chain.

La prima ha il compito di raccogliere le informazioni inserite all'interno degli shard, riassumere lo stato degli stessi, per facilitare la riconciliazione e la trasmissione dati. La seconda assolve al compito di dettare un tempo e un cadenza comune ai diversi shard per la creazione dei nuovi blocchi.

Su queste basi, proponiamo un protocollo di consenso che permetta la realizzazione di blockchain in grado di gestire un alto numero di transazioni al secondo, con una struttura distribuita e sicura. I parametri del protocollo si adattano dinamicamente allo stato della blockchain, permettendo elevata scalabilità.

Indice:

Introduzione

Ogni piattaforma basata sulla tecnologia blockchain, che ambisca a essere utilizzata globalmente, deve essere in grado di gestire un numero sempre maggiore di user e di transazioni da aggiungere a registro, senza andare ad intaccare la fruibilità e la sicurezza del servizio che viene offerto.
Sono state proposte molte soluzioni al problema della scalabilità e molte di queste sono basate sulla tecnica di partizione della blockchain chiamata Sharding [4].

Lo sharding consiste nel separare la blockchain in più catene, indicati con il nome di Shards (frammento, pezzo, scheggia...) o filoni , che vengono gestiti parallelamente da gruppi di nodi appartenenti al network.

Questo approccio aumenta la portata di transazioni gestite dal network di nodi che lavora al consenso stesso. Infatti i nodi, dividendo il lavoro, possono registrare simultaneamente un maggior numero di transazioni.

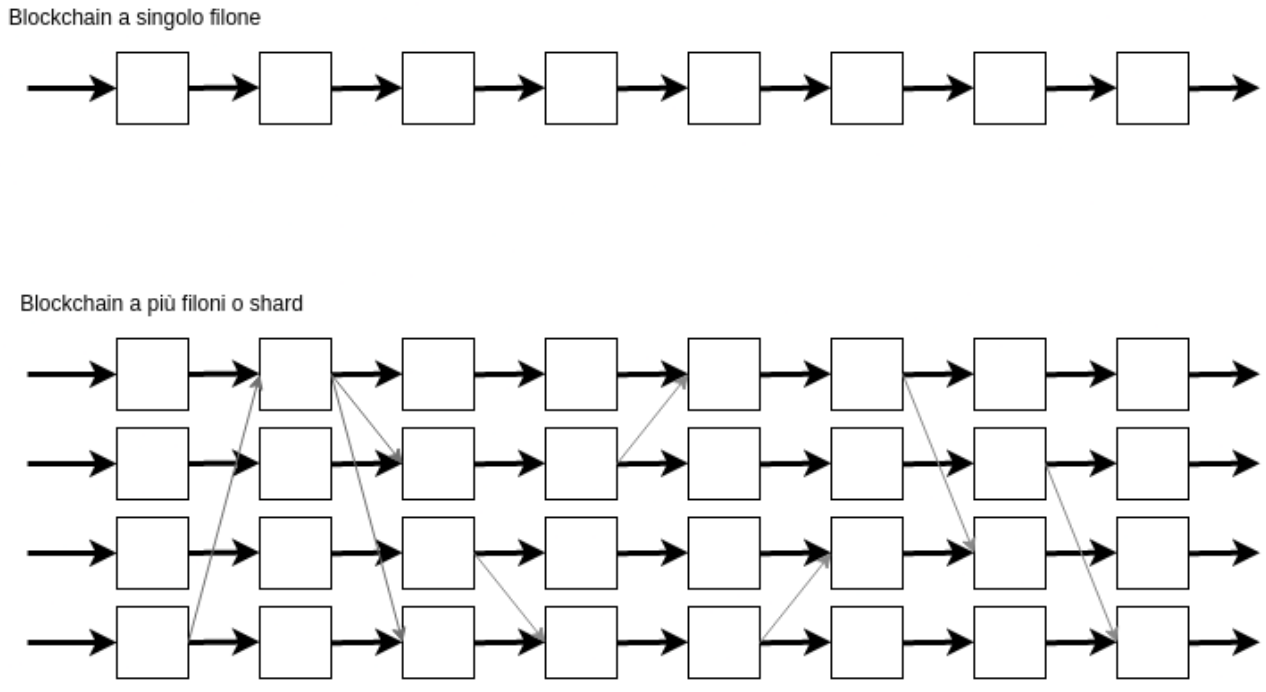

Fig. 1: Differenza tra blockchain monofilare e con sharding

Tuttavia, i vari shard devono essere consistenti tra loro in quanto è il loro insieme che definisce il registro distribuito nella sua interezza.

L'introduzione degli shard ha quindi una contropartita da considerare: le informazioni conservate nel registro devono essere coerenti. Per avere questa coerenza:

- il protocollo di consenso deve essere in grado di gestire anche le comunicazioni tra gli shard,
- vanno considerati i rischi per la sicurezza introdotti da un aumento di complessità del sistema.

Il primo punto ci impone di aggiungere un livello di complessità al protocollo stesso, che si occupi delle comunicazioni tra shard e dei risultati che queste possono portare. Il secondo ci richiede di regolare il lavoro sui singoli shard in modo tale da vincolare l'inserimento di informazioni a registro secondo regole ben definite. Queste regole, pur limitando la portata teorica della blockchain, permettono al network che mantiene la blockchain di difendersi contro attacchi al registro effettuati da nodi malevoli. In particolare, si rende necessario identificare la porzione massima di nodi malevoli sul totale che la blockchain possa tollerare, mantenendo allo stesso tempo le garanzie di incorruttibilità, immutabilità ed affidabilità.

In questo report andremo ad esplorare come, tramite algoritmi di tipo Byzantine Fault Tolerant (BFT), sia possibile progettare protocolli di consenso che implementino la tecnica di Sharding per aumentare la portata di transazioni elaborate, pur gestendo le comunicazioni tra gli shard e mantenendo la sicurezza.

§

Protocollo di Consenso

Un protocollo di consenso è un insieme di istruzioni che i nodi di una rete devono seguire per avere la ragionevole certezza che la loro rete abbia uno stato condiviso. Serve cioè a dare a ogni nodo la garanzia che gli altri conoscano lo stesso insieme di informazioni e dati. Queste informazioni potrebbero essere, ad esempio, il contenuto dei record di un database. In una blockchain, lo stato (contenente tutte le informazioni di interesse comune) si riferisce alla catena di blocchi e alle transazioni in esse contenute.

Si noti che l'adesione al protocollo e il rispetto delle sue regole sono scelte autonome del singolo nodo, soprattutto in una rete Peer-to-Peer permissionless. Non si ha quindi modo di imporre ad un nodo di avere un comportamento corretto rispetto al protocollo, né onesto, né coerente o costante. I nodi il cui comportamento è incerto vengono definiti "bizantini".
In questo documento non faremo assunzioni di alcun genere sul comportamento di un nodo bizantino, neanche quella che ogni nodo persegua il suo interesse personale al meglio delle sue possibilità.

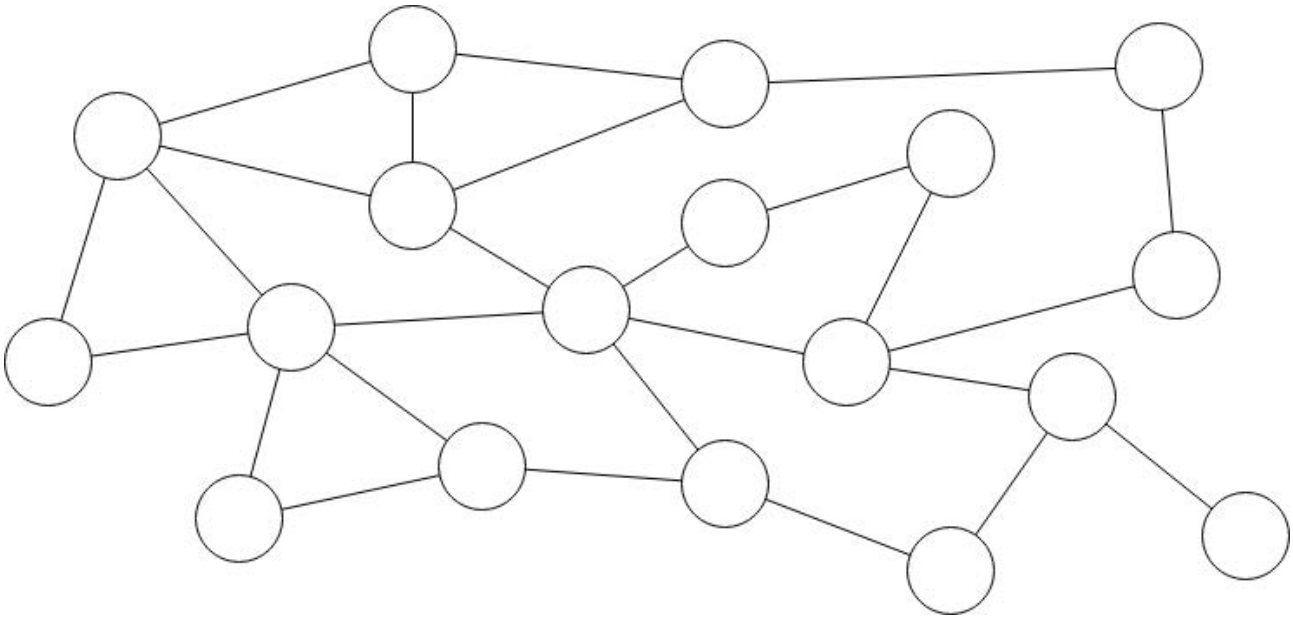

Fig. 2: Rete peer to peer

Scopo dell'algoritmo del consenso quindi è quello di dettare un insieme di regole a cui i nodi aderiscono individualmente e volontariamente al fine di raggiungere il proprio scopo, in modo che le garanzie di allineamento necessarie si abbiano solo seguendo le regole del protocollo. Al tempo stesso si vuole che eventuali danni alla rete derivanti dall'operare di un nodo bizantino siano quanto più limitati e circoscritti. Gli algoritmi del consenso usano una metrica specifica per indicare la frazione di nodi bizantini o in *fault* **f** sul totale di **n** nodi per cui i rimanenti **n-f** nodi onesti possano continuare ad avere le garanzie sul consenso.

§

Algorand

In questa sezione andremo a descrivere il protocollo di consenso della di Algorand [6], una piattaforma che utilizza un protocollo di consenso BFT per aggiornare la Blockchain. In particolare descriveremo quali sono le tre fasi che costituiscono il protocollo di consenso e a cui in seguito faremo riferimento per descrivere una soluzione al problema che verrà descritto nella prossima sezione.

Algorand utilizza delle VRF (verifiable random functions) per selezionare gli account (potential leaders) incaricati di proporre i nuovi blocchi per ogni round. Una volta che i blocchi creati dai potential leaders sono stati proposti al network, dei comitati (selected verifiers) estratti usando una VRF dovranno votare un blocco tra quelli proposti. Se il numero dei membri malevoli appartenenti ai comitati è ridotto, allora si giungerà a un consenso su un nuovo blocco, che verrà chiamato certificato, come spiegato più avanti.

Il protocollo di consenso di Algorand fa parte dei protocolli detti Proof of Stake, in cui il ruolo dei nodi dipende da una qualche quantità posseduta. Nel caso di Algorand, maggiore è il numero di Algos (il nome della criptovaluta) posseduti da un account, maggiore è la probabilità che questo account venga eletto come potential leader oppure come selected verifier.

Riteniamo interessante questo approccio per tre ragioni. Bisogna scegliere comitati ristretti di nodi per garantire che le comunicazioni possano essere eseguite in maniera performante. Questa scelta va fatta per estrazione casuale perché così tutti i nodi hanno la possibilità di essere eletti a prendere parte attivamente al protocollo di consenso, ma anche perché così la percentuale di nodi bizantini nei comitati non si discosta troppo da quella globale sulla rete.

Il protocollo di consenso di Algorando è sommariamente suddiviso in tre fasi: block proposal, soft vote e certify vote.

Ci focalizzeremo principalmente sulle ultime due fasi in quanto saranno quelle che andremo ad adattare alla costruzione del nostro framework. Il framework permetterà di progettare una classe di protocolli di consenso per piattaforme blockchain che implementino lo Sharding.

§

Fase 1 : block proposal

Nella fase di proposta del blocco, ogni nodo **N** nel network esegue una prima VRF con cui verifica se uno degli account che controlla è candidabile a proporre un blocco.
Questo processo richiede di calcolare una funzione nota che si basa sui valori dei precedenti blocchi e di firmare il risultato con la propria chiave privata. Il risultato viene valutato numericamente rispetto ad un valore di soglia per verificare, in modo dimostrabile a terzi, di essere un candidato.

In caso affermativo, **N** raccoglie un insieme di transazioni in stato pending, le valida, le inserisce in un nuovo blocco che diffonde alla rete in modo che ogni altro nodo possa verificarlo. A questo scopo, oltre al nuovo blocco **N** fornisce alla rete tutti i parametri necessari (e.g., la chiave pubblica per verificare la firma) affinché gli altri nodi possano verificare che il mittente sia effettivamente un nodo candidabile, come stabilito dalla VRF.

§

Fase 2 : soft vote

La prima fase, data la natura distribuita della rete e il meccanismo casuale/ probabilistico su cui si basa, può vedere diversi nodi proporre blocchi candidabili legittimi. La fase di soft vote ha quindi come obiettivo quello di filtrare il numero di blocchi proposti fino a ottenere un solo blocco su cui far convergere la scelta.

Ogni nodo del network può ricevere diversi blocchi candidati. Si noti che non è detto che tutti i nodi ricevano lo stesso insieme di blocchi. Per ogni blocco ricevuto quindi il nodo:

1) controlla le credenziali del proponente (autorizzazione della prima VRF),
2) effettua l'hash delle credenziali calcolandone il digest,
3) tratta il digest come un valore numerico,
4) sceglie quella col valore numerico più basso.

Secondo le informazioni che ha ricevuto, ogni nodo è in grado di identificare quale è il leader del round corrente e quindi propaga solo il suo blocco.

A questo punto i nodi del network verificano se uno degli account da loro controllati è stato eletto (via VRF) per esprimere un voto sul blocco da certificare. Questo processo è eseguito utilizzando un protocollo di Graded Consensus [1] alla fine del quale solo un blocco può aver ottenuto i voti necessari per poter portare a termine la terza fase e quindi essere certificato.

§

Fase 3 : certify vote

La terza fase, quella di certificazione, consiste nell'esecuzione di un protocollo di consenso bizantino binario. Se durante il soft vote è stato possibile identificare un blocco come candidato alla certificazione, è possibile che tale blocco venga certificato dagli account dei nodi del comitato. In particolare, se l'agreement è raggiunto sul bit 0 allora il blocco viene aggiunto al registro distribuito, altrimenti, se l'agreement viene raggiunto sul bit 1, allora si ammette il fallimento del raggiungimento del consenso e si aggiunge a registro solo un blocco di default che non contiene alcuna transazione.

§

Definizione del problema

Vogliamo descrivere il design di una classe di protocolli di consenso per piattaforme blockchain che soddisfino le seguenti proprietà:

- implementano lo sharding per essere in grado di supportare un'elevata portata di transazioni;
- la quantità di energia investita dai nodi per mantenere attiva la piattaforma sia ridotta;
- vogliamo fare sì che la creazione del nuovo blocco in ogni catena (Shard o Master-Chain che sia) sia assegnata ad un singolo nodo in modo che i nodi nel network non debbano competere continuativamente tra loro al fine di aggiudicarsi la creazione di un blocco (e.g. Bitcoin [7]), andando quindi a sprecare le risorse investite dai nodi che perderanno la competizione

Per fare ciò, oltre ai singoli shard prenderemo in considerazione una Master-Chain che sarà incaricata di raccogliere le informazioni finalizzabili inserite dai nodi negli Shard. Questo permetterà di semplificare la riconciliazione tra le informazioni presenti sui vari Shard.

Una volta che le informazioni inserite negli Shard vengono raccolte nella Master-Chain ed i blocchi che le contengono vengono finalizzati, non è più necessario tenere traccia delle transazioni anche sugli Shard.

L'assegnazione dei nuovi blocchi da creare ai nodi incaricati può essere eseguita in seguito a competizioni di vario tipo (PoS,PoW,...), per esempio attraverso una votazione su quali possono essere i nodi di cui il network si fida maggiormente, oppure una competizione di tipo proof of work svolta in un ristretto intervallo di tempo (in modo da contenere il consumo energetico).

Per poter rendere attuabile questo tipo di approccio in un ambiente estremamente asincrono diventa necessaria una suddivisione del tempo in intervalli numerabili e ben definiti.

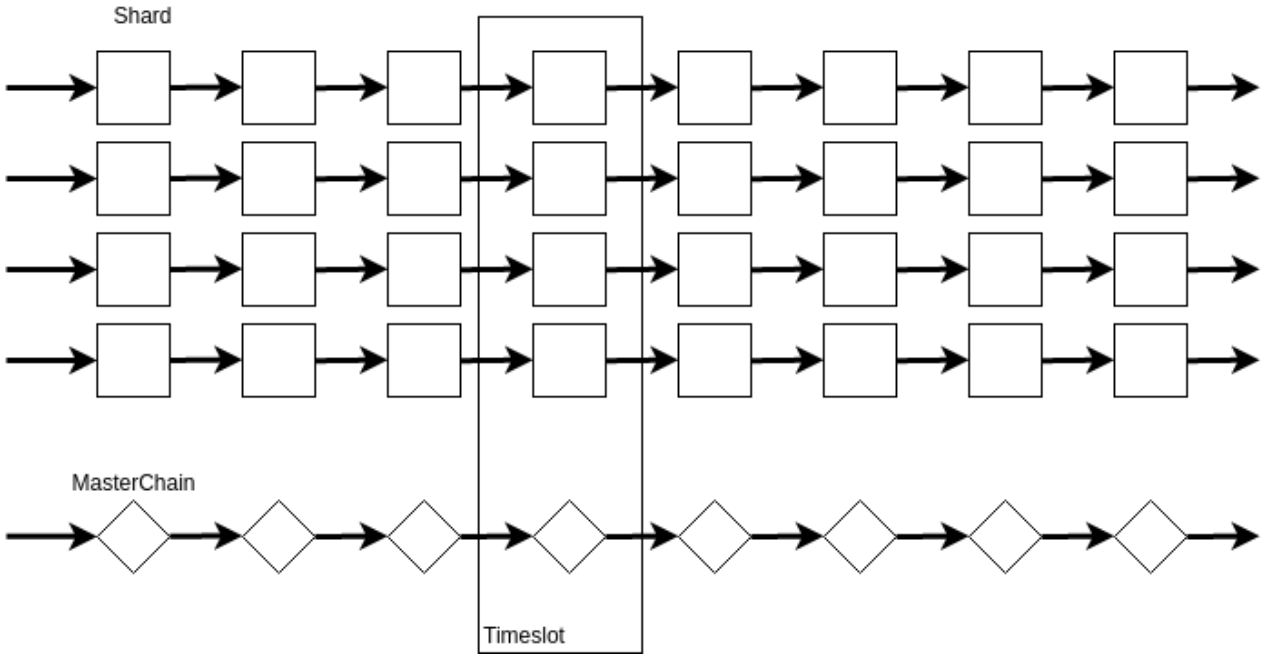

Fig. 3 : Introduzione di time-slot e Master-Chain

Questo aspetto è fondamentale in quanto, per migliorare l'efficienza del protocollo, i nodi incaricati di creare nuovi blocchi devono sapere quando possono iniziare il loro lavoro (ossia quando il blocco precedente al loro è stato creato, condiviso e accettato) ed entro quanto tempo devono condividere con il network il lavoro da loro prodotto. I tempi devono essere scanditi in maniera chiara ed inequivocabile e i blocchi creati dai nodi che non rispettano le regole condivise dal network verranno scartati, andando a perdere gli eventuali reward e/o lo stake che era stato congelato quando il nodo si è candidato per costruire il blocco.

Il problema di progettare un protocollo di consenso che implementi lo Sharding in maniera sostenibile ed efficiente utilizzando l'approccio sopra indicato può essere risolto solo una volta progettato il modo di regolare l'operato dei nodi tramite una suddivisione del tempo in intervalli di lavoro (time-slot).

Nelle sezioni che seguono mostreremo come questo obiettivo può essere raggiunto andando a sfruttare un protocollo di consenso Bizantino parallelizzato che si basa sull'estensione al caso multidimensionale dei sotto-protocolli utilizzati in Algorand, ossia il Graded Consensus [1] ed il Binary Byzantine Agreement [5].

§

Definizione del problema

Descrizione della soluzione

La soluzione che proponiamo è l'introduzione di una nuova catena di blocchi che chiameremo Time-Chain la quale è mantenuta da un insieme specifico di nodi attraverso un protocollo di consenso bizantino. Questo protocollo non prevede la presenza di nodi che svolgono il ruolo di leader andando a proporre blocchi da inserire a registro, bensì la creazione del nuovo blocco avviene in maniera sinergica e parallelamente alla produzione del certificato del blocco stesso.

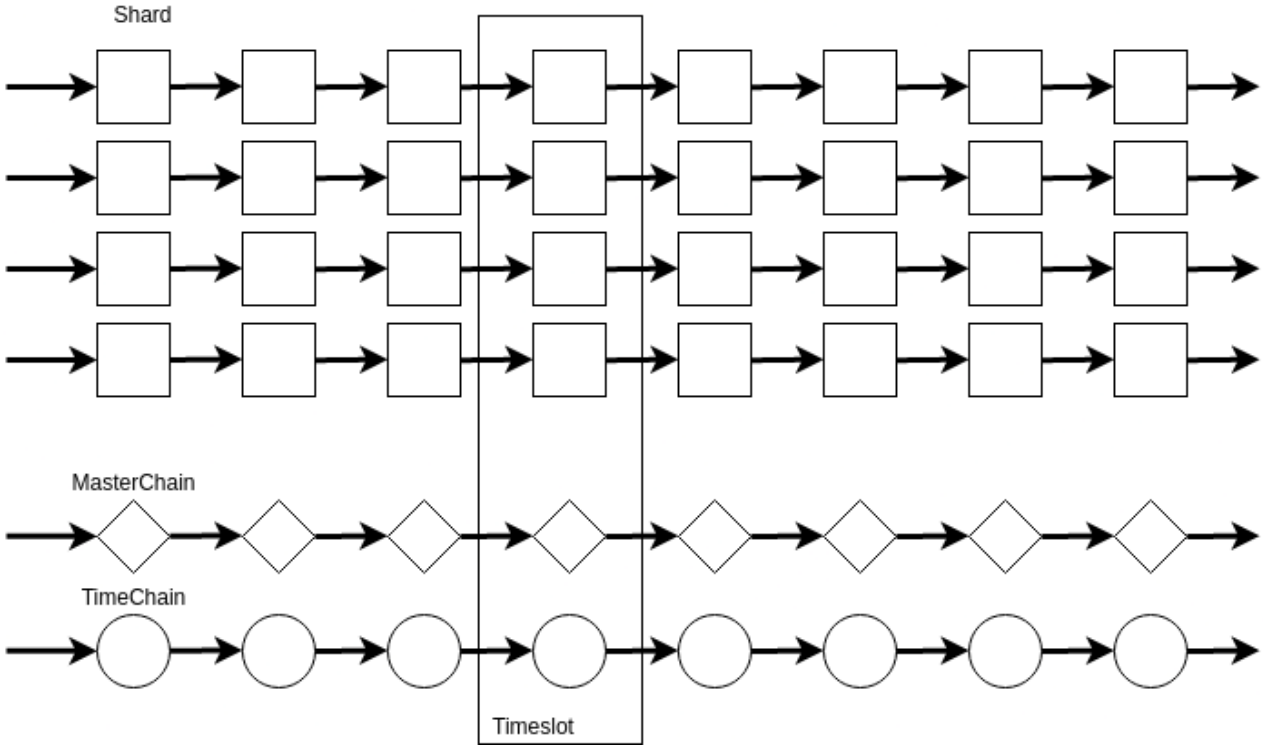

Fig. 4 : Introduzione della Time-Chain

Lo scopo di questa catena di blocchi è quello di scandire l'inizio del nuovo time-slot ad ogni pubblicazione di un nuovo blocco della Time-Chain.

Questo compito non può essere assegnato ai nodi che gestiscono la Master-Chain in quanto questi già devono valutare i blocchi degli Shard da includere nei blocchi della Master-Chain. Inoltre secondo il nostro framework, la creazione dei blocchi della Master-Chain è eseguita da singoli nodi incaricati, per cui un nodo malevolo potrebbe modificare la durata del time-slot o addirittura interrompere la produzione di blocchi e quindi la registrazione delle transazioni. Per questa ragione vogliamo che lo scandire del tempo venga regolato da una moltitudine di nodi attraverso l'esecuzione di un protocollo di consenso bizantino, che renda finale il blocco su cui viene raggiunto il consenso.

Di conseguenza, la soluzione che pare essere più ragionevole è la seguente:

- nello shard al tempo **t** per ogni shard **s**:
  - viene estratto/selezionato un nodo per creare il blocco,
  - viene estratto/selezionato un secondo nodo che parteciperà alla creazione del blocco nella Master-Chain;
- i nodi creano e diffondono i blocchi nei tempi previsti fornendo le necessarie prove (VRF);
- passato il controllo, i digest dei blocchi generati nei diversi shard vengono diffusi ai nodi incaricati della Time-Chain;
- nodi incaricati della Time-Chain :
  - inseriscono il *digest* dei blocchi degli Shard nel blocco della Time-Chain,
  - decretano la fine dell'attuale *time-slot*,
  - e l'inizio del *time-slot* successivo.

Il fatto che il *digest* di un blocco della Master-Chain o di uno Shard sia presente nel corrispondente blocco della Time-Chain sta a significare che il blocco è stato creato del nodo autorizzato e che è stato inviato entro i tempi prestabiliti. Di conseguenza, i nodi che lavorano nel *time-slot* successivo possono esaminare i blocchi del *time-slot* appena concluso e, se li ritengono validi, li prendono in considerazione collegando i blocchi da loro creati a quelli diffusi nel *time-slot* precedente.

All'inizio del t*ime-slot* **t+1** il nodo **n** incaricato di creare il nuovo blocco dello Shard **s** verifica se nel blocco della Time-Chain corrispondente al *time-slot* **t** è presente il *digest* di un blocco dello Shard **s**. Dobbiamo considerare due possibili casi:

1. se il digest non è presente: significa che il blocco non è stato prodotto oppure non è stato diffuso in tempo quindi i nodi che lavorano al consenso sulla Time-Chain l'hanno escluso;
2. se il digest è presente: significa che il blocco è stato creato e diffuso in tempo, quindi il nodo n esegue una valutazione qualitativa del blocco, andando a verificare la validità delle transazioni in esso contenute (verifica che non viene fatta dai nodi che creano i blocchi della Time-Chain). Vanno presi in considerazione due casi:
   a. se il blocco risulta valido, collega il blocco che deve creare al blocco appena verificato,
   b. altrimenti lo scarta collegandosi all'ultimo blocco che ritiene valido nello Shard s.

A sua volta il nodo **n** deve inviare il suo blocco entro un tempo prestabilito durante il *time-slot* **t+1**, in questo modo i nodi che mantengono la Time-Chain possono aggiungere il *digest* del blocco creato nel blocco della Time-Chain a cui lavoreranno per decretare la fine del *time-slot* **t+1** e l'inizio del *time-slot* **t+2**.

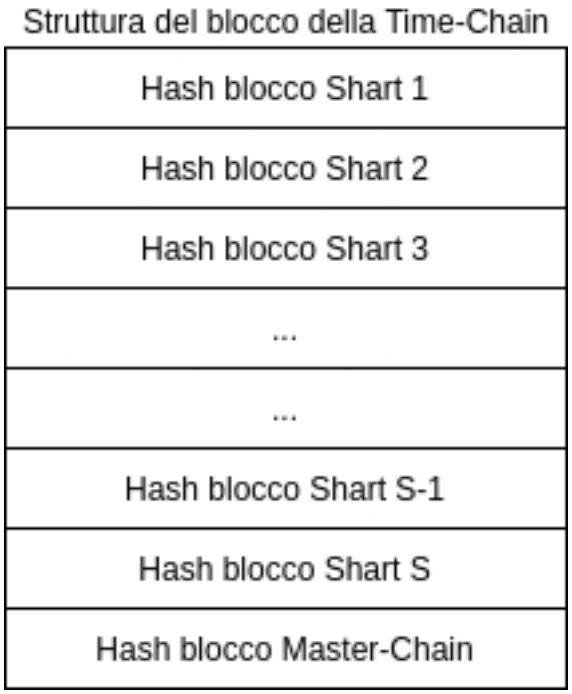

Fig. 5 : Struttura del Blocco della Time-Chain

Di conseguenza possiamo immaginare i blocchi della Time-Chain come vettori di **S+1** componenti dove **S** è il numero di Shard, quindi il vettore ha nella prima componente l'eventuale digest del blocco della Master-Chain (oppure una componente vuota se il blocco non è stato creato o diffuso in ritardo) e nelle altre **S** componenti l'eventuale digest dei blocchi dei vari Shard.

§

Consenso sulla Time-Chain

Concentriamoci dunque su come i nodi che mantengono la Time-Chain possono raggiungere il consenso sui blocchi degli Shard e della Master-Chain da considerare validi, inserendone il digest all'interno del nuovo blocco della Time-Chain.

In un ambiente asincrono e distribuito come quello in esame ogni nodo potrebbe ricevere un pacchetto dati in momenti diversi. Vanno infatti considerati fenomeni come latenza di rete, errori di trasmissione e simili. Di conseguenza è possibile che il messaggio di distribuzione di un nuovo blocco sia ricevuto entro il tempo utile solo da parte dei nodi della rete mentre gli altri lo ricevono fuori tempo massimo. I primi nodi riterranno valido il blocco ricevuto, gli altri lo scarteranno.

Quindi anche assumendo che i nodi che lavorano al consenso sulla Time-Chain siano tutti onesti, è improbabile che tutti i nodi abbiano sempre la stessa opinione su quali blocchi validare e quali scartare. Per di più dobbiamo considerare anche la

presenza di nodi malevoli che cercano di disturbare o impedire il raggiungimento del consenso.

Per costruire un protocollo di consenso solido è necessario che le seguenti proprietà siano soddisfatte:

1) tra i nodi che prendono parte al consenso della Time-Chain una pluralità di essi prende parte alla costruzione del blocco;
2) il processo di consenso viene eseguito parallelamente su ogni componente del vettore.

Le condizioni sono necessarie in quanto la loro assenza potrebbe portare ai seguenti scenari.

Supponiamo l'insieme dei nodi che mantengono la Time-Chain siano i 4 nodi onesti **n1**, **n2**, **n3** e **n4**. Assumiamo inoltre che il numero di Shard sia 3 e che i quattro nodi debbano accordarsi su quali blocchi accettare alla fine di un dato time-slot.

Assumiamo che i digest dei blocchi osservati dai vari nodi (che per semplicità verranno indicati come numeri interi) siano quelli riportati in tabella Tab. 1.

nodo	Master-Chain	Shard1	Shard2	Shard3
n1	9	2	8	4
n2	9	2	7	1
n3	9	3	8	1
n4	0	2	8	1

Tab. 1 :Confronto tra i vettori salvati localmente da ciascun nodo

Possiamo notare che ogni nodo ha ricevuto un'informazione discordante dagli altri nodi in una componente del vettore di digest, quindi tutti i vettori sono diversi tra loro. Possiamo formulare tre ipotesi di possibile risoluzione.

Ipotesi 1:

Se il consenso sul vettore di digest si basasse su un leader (ex: **n3**) che propone un vettore intero (blocco=(**9,3,8,1**)), come per esempio accade nei protocolli di consenso basati su PBFT [3], gli altri nodi non accetterebbero il vettore proposto in quanto diverso da quello che loro hanno osservato (nessuno ha il valore **3** per lo Shard 1).

Di conseguenza in Time-Chain verrebbe aggiunto un blocco di default privo di hash pointer ai blocchi degli Shard e della Master-Chain, causando lo scarto delle informazioni in essi contenute.

Ipotesi 2:

Se il consenso sul vettore di digest si basasse su un leader (ex: **n2**) che propone un vettore (blocco) le cui componento sono valutate singolarmente, gli altri nodi accetterebbero solo 3 componenti su 4 del vettore proposto (**9,2,/,1**) , in quanto la maggioranza dei nodi sarebbe in accordo con 3 componenti proposte dal *leader*.

Di conseguenza, in Time-Chain verrebbe aggiunto un blocco contenente il *digest* di 3 blocchi mentre la componente relativa al valore per cui il leader era in disaccordo con gli altri nodi verrebbe scartata.

Ipotesi 3:

Se il consenso sul vettore di *digest* prevedesse la condivisione dei valori osservati da parte di più nodi e la valutazione venisse fatta singolarmente su ogni componente, sarebbe possibile inserire nel blocco della Time-Chain il vettore contenente i valori (**9,2,8,1**) in quanto per ogni valore la maggioranza dei nodi sarebbe in accordo.
Questo approccio restituisce, in questo esempio, un risultato massimale in termini di informazioni registrate senza centralizzare la decisione un singolo nodo, ma distribuendo la responsabilità su un gruppo.

Evidentemente in un contesto in cui il numero di nodi può essere elevato, ascoltare l'opinione di tutti i nodi non è sostenibile a causa dei tempi di diffusione dei messaggi e del costo in termini di banda per gestire una comunicazione così pesante. Di conseguenza va favorito un approccio che si basa sull'estrazione casuale di un numero supportabile di nodi che comunichi al network la propria opinione riguardo alla valorizzazione delle componenti del vettore.

§

Descrizione dell'implementazione

Andiamo quindi a descrivere la struttura del protocollo in questione. Le specifiche del protocollo si trovano in [2].

Il protocollo prevede una regola di selezione in base alla quale i diversi nodi della rete possono essere selezionati per fare parte dei Time-Nodes. Questi sono l'insieme dei nodi che possono partecipare al protocollo di consenso per l'aggiunta di blocchi alla Time-Chain. La regola di selezione può avere forme diverse per i singoli casi. Alcuni esempi possono essere:

- nodi che controllano chiavi private con un saldo superiore ad un valore di soglia,
- nodi selezionati tramite una VRF.

Ad ogni *time-slot*, un sottoinsieme dei Time-Nodes viene estratto casualmente tramite VRF per creare il prossimo blocco della Time-Chain a partire dai dati dei blocchi precedenti.

§

Multidimensional Graded Consensus Protocol

Il sottoinsieme dei Time-Nodes incaricati di diffondere i messaggi dello STEP 1 costruiscono il vettore contenente il digest dei blocchi che hanno ricevuto entro i tempi prestabiliti e lo diffondono. Utilizzando quindi una variante parallelizzata del protocollo Graded Consensus [1] su ogni componente del vettore, in tre step ogni nodo n è in grado di ricostruire un vettore $\mathbf{v}_n$ contenente in ogni componente il valore che sarà inserito nel blocco della Time-Chain se quella componente dovesse essere valorizzata.

NB: se due nodi onesti (ossia che hanno seguito fedelmente il protocollo) alla fine del Multidimensional Graded Consensus Protocol hanno salvato localmente due valori distinti nella stessa componente, allora non è possibile che quella componente venga valorizzata nel nuovo blocco della Time-Chain. Infatti le componenti che vengono valorizzate sono al più quelle su cui tutti gli onesti sono in accordo.

nodo	Master-Chain	Shard1	Shard2	Shard3
onesto n1	9	2	8	4
onesto n2	9	2	9	4
malevolo n3	?	?	?	?
onesto n4	9	3	2	4
malevolo n5	?	?	?	?
onesto n6	9	2	4	4
onesto n7	9	2	7	4

Tab. 2 :valori salvati localmente dai nodi appartenenti al network

La tabella Tab. 2 mostra i vettori salvati localmente da ogni nodo alla fine dei tre step di Multidimensional Graded Consensus e in particolare notiamo che, poiché sullo Shard 1 e 2 esistono due nodi onesti che hanno salvato localmente due valori distinti, allora, per le proprietà del protocollo in questione, è impossibile che l'output del protocollo di consenso abbia le corrispondenti componenti valorizzate.

Non ci interessiamo dei valori salvati localmente dai nodi malevoli in quanto questi potrebbero non seguire le indicazioni prescritte nel protocollo di consenso.

In questo esempio l'agreement alla fine del protocollo di consenso può essere raggiunto su uno dei seguenti vettori: (**9** , ***l*** , ***l*** , **4**), (***l*** , ***l*** , ***l*** , **4**), (**9** , ***l*** , ***l*** , ***l***) o (***l*** , ***l*** , ***l*** , ***l***) dove il simbolo ***l*** è usato per indicare quelle componenti su cui non è stato possibile raggiungere il consenso su un valore (non nullo) e quindi sono state lasciate vuote.

§

Multidimensional Binary Byzantine Agreement Protocol

Una volta che ogni nodo **n** ha costruito il proprio vettore $\mathbf{v}_n$ dei possibili valori per il blocco della Time-Chain, i nodi onesti devono accordarsi su quali sono le componenti che hanno in comune e scartare quelle in cui differiscono.

Questo processo può essere svolto utilizzando una variante parallelizzata sulle componenti del vettore di digest del protocollo *Binary Byzantine Agreement* [5] in cui i nodi estratti casualmente per diffondere i messaggi ad ogni step diffondono vettori di bit invece che vettori di digest andando ad alleggerire il peso dei messaggi e quindi ridurre il corrispondente tempo di diffusione. In particolare il bit 1 rappresenta l'intenzione di scartare una componente, mentre lo 0 rappresenta l'intenzione di mantenerla.

Sulla base dei messaggi ricevuti negli step di Multidimensional Graded Consensus, i nodi sono in grado di ricostruire, oltre che $\mathbf{v}_n$, un vettore di bit in cui in ogni componente viene registrato 0 se il nodo è convinto che tutti i nodi onesti abbiano valorizzato allo stesso modo quella componente, altrimenti viene registrato 1. I nodi incaricati diffonderanno il vettore di bits dando inizio all'esecuzione del protocollo *Multidimensional Binary Byzantine Agreement* raggiungendo, dopo un numero di step non determinabile a priori, il consenso su un vettore di bit.

Se l'agreement viene raggiunto nella componente c sullo 0 significa che tutti i nodi onesti hanno salvato, alla fine del Multidimensional Graded Consensus, lo stesso digest nella **c**-esima componente del vettore $\mathbf{v}_n$. Se invece il consenso viene raggiunto su 1 allora quella componente verrà scartata e il relativo Shard (o Master-Chain se **c**=1) non avrà nessun blocco aggiunto in coda.

In particolare, seguendo l'esempio della tabella Tab. 2, l'agreement può essere raggiunto sui vettori di bit (**0,1,1,0**), (**1,1,1,0**), (**0,1,1,1**) oppure (**1,1,1,1**).

Indipendentemente su quale dei quattro vettori i nodi raggiungono il consenso, questi possono ricostruire un unico e comune vettore di digest da inserire a registro come è mostrato in tabella *Tab. 3*.

vettore di bit accordato	(0,1,1,0)	(1,1,1,0)	(0,1,1,1)	(1,1,1,1)
corrispondente vettore di hash	(9,/,/,4)	(/,/,/,4)	(9,/,/,/)	(/,/,/,/)

Tab. 3 : Ricostruzione del vettore di digest sulla base del vettore di bit su cui è stato raggiunto il consenso

Infatti, per come sono stati progettati i protocolli, non è possibile raggiungere l'agreement su 0 nelle componenti relative a Shard 1 e Shard 2 dell'esempio, in quanto esistono, per ciascuna componente, due nodi onesti che hanno salvato valori distinti.
Sarà invece possibile, ma non è garantito, che l'agreement venga raggiunto su 0 nelle componenti corrispondenti allo Shard 4 e alla Master-Chain. Queste proprietà sono formalmente dimostrate in [2].

Questo protocollo tuttavia è un protocollo iterativo ed il numero di step richiesti per raggiungere l'agreement sul vettore di bit dipende dal numero di componenti del vettore e dalla frazione di nodi malevoli nel network.
Si possono dunque scegliere, a seconda dello use case e dalle caratteristiche dei protocolli di consenso sugli Shard e sulla Master Chain, due possibili approcci:

1. cercare di inserire nel blocco della Time-Chain più digest possibili a discapito di una precisa costanza nei tempi di creazione dei blocchi della Time-Chain;
2. scartare le componenti su cui non si riesce a raggiungere il consenso (quindi settarle a 1 nel vettore di bit) garantendo la terminazione del protocollo entro un numero fissato di step.

§

Termination Steps

Se si sceglie il primo approccio allora il protocollo termina non appena il consenso viene raggiunto sul vettore di bit che comunica ai nodi quali componenti valorizzare e quali no. Se si sceglie il secondo approccio invece bisogna definire un terzo sotto-protocollo che permetta ai nodi onesti di capire quali sono le componenti su cui non è stato raggiunto il consenso.
Per evitare attacchi che si basano sul convincere alcuni nodi della necessità di scartare una componente e convincere altri del raggiungimento del consenso, si esegue un numero K di step che rendono probabilisticamente infattibile questi tipi di attacco, impedendo dunque la creazione di due certificati per due blocchi distinti della Time-Chain.

§

Risultati ottenuti

Sfruttando l'architettura descritta nelle sezioni precedenti, è possibile regolare una piattaforma blockchain che implementa lo sharding e che possa garantire un'elevata portata di transazioni e richieste a smart contracts. Alla base di questo approccio nel regolamentare l'inserimento a registro delle transazioni troviamo la suddivisione del tempo in intervalli di lavoro ben definiti. Infatti, in ogni time-slot i nodi incaricati di aggiornare il registro (sugli Shard oppure sulla Master-Chain) possono fare affidamento sulle informazioni inserite a registro fino a quel momento, consapevoli del fatto che non cambieranno per tutta la durata del time-slot corrente.

La suddivisione del tempo avviene sfruttando un protocollo di consenso BFT che va a certificare il lavoro svolto dai nodi che hanno lavorato correttamente.
I blocchi creati utilizzando protocolli BFT sono finali nel momento in cui vengono creati, di conseguenza è possibile andare a coordinare il lavoro dei nodi nonostante le comunicazioni avvengano in modo asincrono. Infatti, assumendo che il tempo di diffusione di un blocco della Time-Chain sia minore di un certo parametro **T**, sappiamo che l'inizio o la fine di un dato time-slot per due nodi distinti è minore o uguale a **T**. Questo ci basta per sincronizzare il lavoro dei nodi nel network permettendo la parallelizzazione del lavoro attraverso la tecnica di Sharding.

Il protocollo di consenso sulla Time-Chain descritto nelle sezioni precedenti garantisce il raggiungimento del consenso in un numero finito di step, di conseguenza si può definire un upper-bound temporale per la creazione di ogni blocco. Tutto questo è reso possibile anche dal fatto che il protocollo è leaderless, di conseguenza ogni blocco certificato viene creato sinergicamente dai nodi che prendono parte al protocollo di consenso. A differenza dei protocolli che si basano su PBFT [3], che prevedono la possibilità di ricominciare un'esecuzione del protocollo in quanto viene provato che il leader è malevolo oppure non attivo, nel nostro caso ogni esecuzione del protocollo verrà portata a termine con la generazione di un blocco contenente informazioni significative e finali.

Questa proprietà offre la garanzia ai nodi che lavorano sugli Shard e sulla Master-Chain che, eseguono il lavoro entro i tempi prestabiliti, il loro blocco supererà il primo step di validazione, e che la validazione stessa non è nelle mani di un solo nodo che potrebbe intenzionalmente vanificare il lavoro da loro svolto.

Infine, un'ulteriore aspetto che garantisce l'efficienza e la scalabilità della classe di protocolli che abbiamo descritto è il fatto che, indipendentemente dal numero di shard attivi in un dato time-slot, il protocollo di consenso bizantino, che produce la validazione preliminare dei blocchi inseriti a registro, richiede un numero approssimativamente costante di messaggi diffusi nel network. Infatti, al crescere degli shard il protocollo di consenso può richiedere più step per raggiungere il consenso, tuttavia l'upper-bound $S+K$ di step non può essere superato per definizione del protocollo. Inoltre, il numero di messaggi diffuso dipende solo dal

numero di nodi incaricati di diffondere il messaggio ad ogni round ed il numero di shard impatta solamente sul numero di componenti del vettore (di digest nei primi due step e di bit in tutti gli step successivi) contenuto in ogni messaggio.
Nonostante il peso di ogni messaggio aumenti al crescere del numero di shard, questo incremento in peso è irrilevante rispetto al peso delle firme digitali necessarie a ciascun nodo per dimostrare di essere incaricati a diffondere il proprio messaggio.

❖ ❖ ❖

Bibliography

[1] Pesech Feldman and Silvio Micali. An optimal probabilistic protocol for synchronousbyzantine agreement.SIAM Journal on Computing, 26(4):873–933, 1997.

[2] Flamini, A. (2021). A Byzantine Fault Tolerant Consensus Protocol for Parallel Time-Stamping. Università di Trento. https://webapps.unitn.it/Biblioteca/it/Web/Tesi

[3] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In OSDI, volume 99, pages 173–186, 1999.

[4] Meneghetti, A., Parise, T., Sala, M., & Taufer, D. (2019). A survey on efficient parallelization of blockchain-based smart contracts. arXiv preprint arXiv:1904.00731.

[5] Silvio Micali. Byzantine agreement, made trivial, 2016.

[6] Jing Chen and Silvio Micali. Algorand: A secure and efficient distributed ledger.Theoretical Computer Science, 777:155–183, 2019

[7]Satoshi Nakamoto. Bitcoin whitepaper. 2008.