
This is the **published version** of the bachelor thesis:

Kochenov, Nikita. *Sistema distribuït de versionat amb contribucions anònimes verificables*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Enginyeria Informàtica)

This version is available at <https://ddd.uab.cat/record/334374>

under the terms of the  license.

Sistema de versionat distribuït amb contribucions anònimes verificables

Nikita Kochenov

Juny de 2026

Resum— Aquest treball presenta el disseny i la implementació d'un prototip de sistema distribuït de control de versions que permet a un conjunt definit de membres realitzar contribucions criptogràficament verificables sense revelar quina identitat concreta n'és l'autora. El sistema es construeix sobre una xarxa P2P amb `libp2p` i adopta un model de dades en graf acíclic dirigit (DAG) similar al de Git, ampliat amb el concepte de *RingVersion* per representar la composició del grup de possibles contribuïdors al llarg del temps. Els commits se signen mitjançant l'esquema de signatures d'anell de Rivest, Shamir i Tauman (RST) sobre claus RSA-2048, garantint anonimat incondicional i exculpabilitat. Un mecanisme de govern distribuït permet modificar la composició del grup mitjançant *proposals* de canvi que requereixen l'aprovació d'una supermajoria de membres ($\lfloor 2n/3 \rfloor$), signades individualment amb una variant probabilística de l'RSA, sense necessitat d'autoritat central. El prototip s'ha validat en un entorn de tres nodes amb Docker Compose, comprovant la propagació de commits, la verificació de signatures i el rebuig de contribucions invàlides. La conclusió principal del treball és que l'anonimat incondicional i la responsabilitat individual no poden coexistir dins d'un mateix esquema criptogràfic: els problemes de governança d'un sistema anònim han de resoldre's necessàriament fora del propi sistema criptogràfic.

Paraules clau— Sistemes de control de versions distribuïts, signatures d'anell, criptografia RSA, anonimat verificable, xarxes P2P, `libp2p`, governança distribuïda, tolerància a fallades bizantines.

Abstract— This work presents the design and implementation of a prototype distributed version control system that allows a defined set of members to make cryptographically verifiable contributions without revealing the specific identity of the author. The system is built on a P2P network using `libp2p` and adopts a directed acyclic graph (DAG) data model similar to Git's, extended with the concept of *RingVersion* to represent the composition of a group of possible contributors over time. Commits are signed using the Rivest–Shamir–Tauman (RST) ring signature scheme over RSA-2048 keys, providing unconditional anonymity and exculpability. A distributed governance mechanism allows the group membership to evolve through change proposals that require approval from a supermajority of members ($\lfloor 2n/3 \rfloor$), individually signed with a probabilistic variant of RSA, without any central authority. The prototype was validated on a three-node Docker Compose environment, verifying commit propagation, signature verification, and rejection of invalid contributions. The main conclusion is that unconditional anonymity and individual accountability cannot coexist within the same cryptographic scheme: governance problems in an anonymous system must necessarily be resolved outside the cryptographic layer itself.

Keywords— Distributed version control systems, ring signatures, RSA cryptography, verifiable anonymity, P2P networks, `libp2p`, distributed governance, Byzantine fault tolerance.



-
- E-mail de contacte: nikita.kochenov@autonoma.cat
 - Menció: Tecnologies de la Informació
 - Treball tutoritzat per: Cristina Pérez Solà (Departament d'Enginyeria de la Informació i de les Comunicacions)
 - Curs 2025/26

1 INTRODUCCIÓ

ELS sistemes de control de versions distribuïts (DVCS), com Git [1], han esdevingut l'eix del desenvolupament de programari col·laboratiu modern. Garanteixen integritat i traçabilitat de l'historial mitjançant funcions hash i associen de manera obligatòria cada contribució a una identitat persistent. Aquesta vinculació identitat–contribució és essencial per a la responsabilitat i l'auditabilitat, però no contempla escenaris en què es vulgui dissociar el contingut aportat de la identitat individual, mantenint alhora garanties criptogràfiques de pertinença a un grup tancat.

Aquesta necessitat pot sorgir, per exemple, en projectes col·laboratius on els participants vulguin reportar problemes o aportar millores sense revelar-se dins del grup, en entorns d'investigació on es vulguin evitar biaixos d'autoria, o en comunitats on la privacitat sigui un requisit, sense renunciar a la verificació de la legitimitat de les aportacions.

1.1 Plantejament del problema

Els DVCS actuals proporcionen un model robust de traçabilitat: cada versió s'associa explícitament al seu autor. La introducció d'anonimat planteja tres reptes: (1) garantir que només membres autoritzats generen contribucions vàlides, (2) mantenir la integritat de la cadena de versions sense autoritat central, i (3) contemplar mecanismes de governança que permetin, sota consens col·lectiu, modificar la composició del grup de membres autoritzats. La hipòtesi central d'aquest treball és que és possible combinar anonimat, verificabilitat criptogràfica i governança col·lectiva en un sistema distribuït de control de versions, sense recórrer a infraestructura centralitzada.

1.2 Objectius

Els objectius generals són: (1) dissenyar i implementar un sistema distribuït de control de versions que permeti contribucions anònimes verificables dins d'un conjunt definit de membres, i (2) analitzar les propietats de seguretat del sistema, amb especial atenció a l'anonimat verificable, la gestió descentralitzada de claus i els mecanismes de validació col·lectiva.

Els objectius específics són: (a) analitzar els mecanismes existents de DVCS i les tècniques criptogràfiques orientades a anonimat i verificabilitat; (b) definir un model de sistema que especifiqui l'arquitectura, el model d'amenaça i les propietats de seguretat requerides; (c) dissenyar un mecanisme que permeti a membres autoritzats realitzar contribucions verificables sense revelar la seva identitat individual; i (d) implementar un prototip funcional que materialitzi el model proposat.

La resta del document s'organitza com segueix: la secció 2 revisa l'estat de l'art en DVCS i criptografia per a l'anonimat; la secció 3 descriu la metodologia seguida; la secció 4 detalla el disseny i la implementació del sistema; la secció 5 presenta els resultats de validació; i la secció 6 en discuteix les conclusions i les línies futures.

2 ESTAT DE L'ART

2.1 Sistemes de control de versions distribuïts

Els DVCS es basen en un model de dades orientat a grafs, on l'historial es representa com un graf acíclic dirigit (DAG): cada node és un commit identificat pel hash criptogràfic del seu contingut i de les referències als seus pares, garantint que qualsevol modificació invalida transitivament els descendents. Cada participant manté una còpia completa del repositori, eliminant el punt únic de fallada propi dels sistemes centralitzats, i la sincronització es realitza mitjançant intercanvi selectiu d'objectes (*push/pull*). La creació concurrent de versions pot generar branques divergents, de manera anàloga al model de replicació multi-mestre i consistència eventual de sistemes com Bayou [3].

Tot i la seva arquitectura distribuïda, estudis empírics mostren que la majoria de projectes amb DVCS adopten fluxos de treball lògicament centralitzats, amb un repositori de referència i un conjunt reduït de mantenidors [1, 2]. Els mecanismes tradicionals tampoc estan dissenyats per proporcionar anonimat dins del conjunt de col·laboradors ni validació criptogràfica basada exclusivament en la pertinença a un grup.

2.2 Primitives criptogràfiques bàsiques

Les funcions hash criptogràfiques i les estructures encadenades [4, 5] proporcionen integritat: qualsevol modificació del contingut altera l'identificador resultant. Tanmateix, la integritat no implica autenticitat. Els esquemes de signatura digital estàndard [6] permeten que qualsevol tercer verifiqui, mitjançant la clau pública de l'emissor, que un missatge prové del posseïdor de la clau privada corresponent, establint cercles de confiança basats en identitats verificables –però sempre vinculant la contribució a una identitat concreta.

2.3 Signatures d'anell i anonimat de conjunt

Les signatures d'anell, introduïdes per Rivest, Shamir i Tauman (RST) [7], permeten que un membre d'un conjunt de claus públiques (l'*anell*) signi un missatge demostrant pertinença al grup sense revelar quina clau privada concreta s'ha emprat. La verificació només garanteix que la signatura prové d'algun membre de l'anell; totes les claus incloses són computacionalment indistingibles respecte de l'origen real, i l'anell es construeix *ad hoc* sense necessitat de configuració prèvia ni d'una entitat gestora.

Aquesta propietat és atractiva per demostrar que un commit prové d'un membre autoritzat sense revelar-ne la identitat. Tanmateix, l'anonimat que ofereix RST és **total i irrevocable**: no existeix cap mecanisme criptogràfic d'obertura posterior ni de revocació dinàmica de membres, cosa que pot ser problemàtica en un sistema amb necessitats d'auditoria.

Una extensió rellevant és el model de signatures d'anell vinculables (*Linkable Spontaneous Anonymous Group*, LSAG) de Liu, Wei i Wong [8], on cada signatura inclou una etiqueta de vinculació derivada de la clau privada del signant que permet detectar si dues signatures provenen del

mateix membre, sense revelar-ne la identitat, i establir-ne l'autoria en cas de *subpoena* de la clau privada. No obstant això, la introducció de vinculabilitat trenca l'anonimat incondicional: l'anonimat passa a ser condicionalment revocable, i qualsevol mecanisme que permeti identificar un autor en determinades circumstàncies debilita estructuralment les garanties d'anonimat per a tots els casos.

2.4 Signatures de grup amb obertura i revocació

El model de signatura de grup de Chaum i van Heyst [9] introdueix un *group manager* que, durant una fase de *setup*, genera una clau pública de grup i claus mestres, i que pot, mitjançant un procediment d'*open*, determinar l'autor d'una signatura concreta i revocar-lo del grup. Aquest model ofereix un equilibri entre anonimat i responsabilitat, però requereix una entitat central que generi i redistribueixi claus privades en cada canvi de composició del grup —una dependència incompatible amb una arquitectura P2P homogènia sense nodes privilegiats, i que exposaria les identitats dels membres durant la redistribució.

2.5 Criptografia lliardar i tolerància a fallades bizantines

Els esquemes de criptografia lliardar [10, 11] distribueixen una capacitat (per exemple, la d'obertura d'una signatura) entre un subconjunt de membres, de manera que cal que un nombre mínim hi participi per executar l'operació. Aquest enfocament és coherent amb el problema dels generals bizantins [13], que estableix que, per garantir consens correcte en presència de membres maliciosos o particions de xarxa, cal una majoria qualificada de $\lfloor 2n/3 \rfloor$ participants honestos. Aquest principi fonamenta el mecanisme de quòrum del sistema de govern descrit a la secció 4.6.

2.6 Treballs relacionats

Les aproximacions existents a l'anonimat en sistemes de control de versions operen en un registre diferent al d'aquest treball. Eines com Gitmask o Anonymous GitHub [14] eliminen metadades identificatives dels commits (nom, correu, marca temporal) o actuen com a intermediaris davant plataformes centralitzades; l'anonimat és purament operacional i depèn de confiar en un servei extern que, en tot moment, coneix la identitat real de l'autor.

Les signatures d'anell s'han aplicat extensament en sistemes de pagament privats com Monero, que empra l'esquema LSAG [8] per anonimitzar transaccions en un registre públic obert. Tanmateix, aquest model difereix substancialment del proposat en aquest treball: l'anell és obert i qualsevol pot formar-ne part, no existeix un model de dades d'historial de versions, i la governança del grup no és un objectiu del sistema.

No s'ha identificat cap proposta publicada que combini les tres propietats simultàniament: anonimat criptogràficament garantit (no operacional), dins d'un grup tancat amb composició gestionada col·lectivament, integrat en un model de control de versions amb estructura DAG. Aquest treball aborda precisament aquest espai.

3 METODOLOGIA

La metodologia segueix un procés iteratiu, similar als models seqüencials clàssics d'enginyeria del programari [12], organitzat en quatre fases que es retroalimenten: (1) *estudi de l'estat de l'art*, centrat en DVCS i primitives criptogràfiques per a l'anonimat verificable; (2) *disseny conceptual*, en què es plantegen models d'arquitectura preliminar i es validen lògicament, revisant-se les fonts quan apareixen inconsistències; (3) *implementació iterativa del prototip* en Go, desenvolupant components en seccions lògiques que es validen abans d'integrar-se; i (4) *avaluació i validació*, comprovant el funcionament dels mecanismes implementats i les propietats de seguretat establertes.

El prototip s'ha desenvolupat en Go, utilitzant `libp2p` per a la capa de xarxa P2P i `Docker Compose` per simular un entorn multi-node controlat. Sempre que ha estat possible s'han emprat llibreries criptogràfiques estàndard de la biblioteca de Go en lloc d'implementacions pròpies, prioritzant un prototip funcional amb les propietats essencials per sobre de la cobertura completa de funcionalitats.

4 DISSENY I IMPLEMENTACIÓ DEL SISTEMA

Abans d'entrar en el detall de cada mecanisme, convé introduir quatre conceptes propis del sistema que apareixen de manera transversal a les seccions següents.

Blobs. El contingut dels fitxers s'emmagatzema separatament dels metadades del commit, identificat pel hash SHA-256 del seu contingut. Un commit només conté una referència a aquest hash.

Epoch i RingVersion. Com que la composició del grup pot canviar al llarg del temps, cada estat del grup s'identifica amb un número d'*Epoch* i es representa com una *RingVersion*: la llista de membres actius i les seves claus públiques RSA en aquell moment. Cada commit queda associat a l'*Epoch* vigent en el moment de signar-se, de manera que la verificació sempre es realitza contra la composició del grup correcta.

Proposals. Com que RST no permet identificar l'autor d'un commit, qualsevol canvi en la composició del grup requereix consens col·lectiu. Una *proposal* és una proposta de nova *RingVersion* que s'activa automàticament quan acumula una supermajoria de $\lfloor 2n/3 \rfloor$ vots vàlids dels membres actuals.

Registre de claus. El `KeyRegistry` és el registre local de claus públiques RSA dels membres, reconstruït a partir de les *RingVersions* persistides en cada arrencada, sense servidor central de claus.

4.1 Arquitectura general

El sistema adopta una arquitectura P2P homogènia: tots els nodes tenen el mateix rol i no existeix cap entitat central de coordinació. Cada peer manté el seu propi estat local, descobreix altres nodes automàticament dins la xarxa local mitjançant `libp2p` (mDNS [17], *Multicast DNS*, un protocol de descobriment automàtic de serveis en xarxa local), i estableix connexions directes que desencadenen els protocols resumits a la Taula 1.

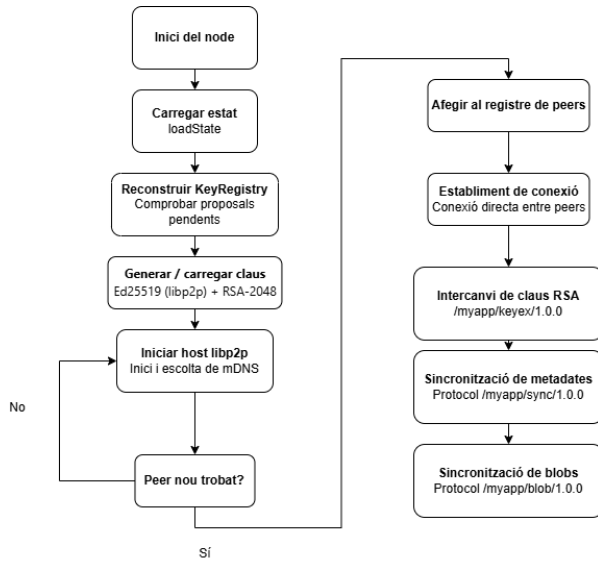


Fig. 1: Flux d'arrencada d'un node: fases d'inicialització, descobriment de peers i handshake.

TAULA 1: PROTOCOLS `LIBP2P` IMPLEMENTATS

Protocol	Funció
/myapp/sync/1.0.0	Sincronització bidireccional de metadades (projectes, membres, <i>RingVersions</i> , commits).
/myapp/commit/1.0.0	Propagació per <i>gossip</i> d'un commit nou als membres connectats.
/myapp/blob/1.0.0	Petició/resposta de <i>blobs</i> absents (<i>pull</i>).
/myapp/blobpush/1.0.0	Enviament proactiu (no sol·licitat) d'un <i>blob</i> , per a deniability plausible.
/myapp/keyex/1.0.0	Intercanvi de claus públiques RSA entre peers.
/myapp/proposal/1.0.0	Propagació i votació de propostes de canvi de <i>RingVersion</i> .

4.2 Model de dades

L'estat es representa jeràrquicament amb tres nivells: *Projecte* (amb identificador, nom, llista de membres i fitxers), *Fitxer* (nom immutable i seqüència de commits) i *Commit* (identificat pel hash del seu contingut, amb *ParentIDs*, *timestamp*, missatge, referències a *blobs*, *Epoch* i *Signature*), formant un DAG anàleg al de Git.

Per suportar canvis en la composició del grup al llarg del temps, s'introdueix el concepte de *RingVersion*: una estructura que representa la composició del grup en un *Epoch* determinat, contenint la llista de *Members* (cada un amb el seu *PeerID* i la seva clau pública RSA). Cada commit queda associat a l'*Epoch* vigent en el moment de signar-se, i la verificació de la seva signatura es realitza contra la *RingVersion* corresponent.

4.3 Identitat criptogràfica i gestió de claus

`libp2p` gestiona internament un parell de claus Ed25519[16] per node (persistides en

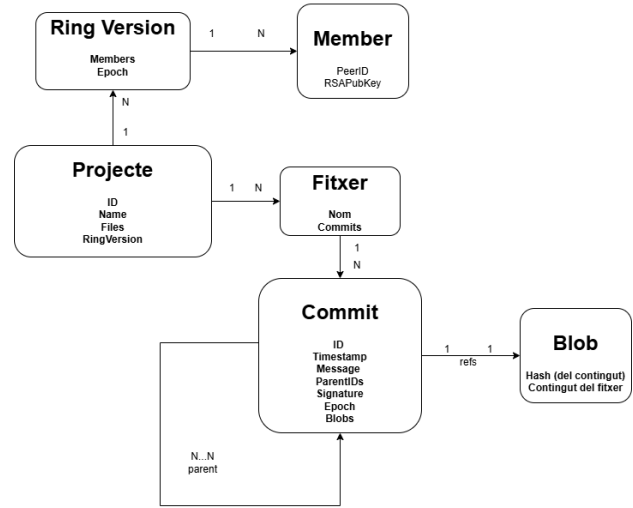


Fig. 2: Model de dades del sistema: relacions entre Projecte, Fitxer, Commit, Blob, RingVersion i Member.

/data/host_identity.key), emprades per a l'autenticació de connexions i el transport segur. El sistema de versionat requereix un segon parell de claus, de tipus RSA-2048[15], generat en el primer arrencament i persistit a /data/identity.key sense regenerar-se en arrencades posteriors.

Aquesta separació respon a dos motius. El primer és el desacoblament de capes: un compromís de la capa de transport no hauria d'afectar la integritat de l'historial de versions, i viceversa. El segon és estructural: l'esquema RST (descriu posteriorment a la secció 4.4) requereix específicament la permutació trampa de RSA, $g_i(x) = x^e \bmod n_i$, invertible únicament pel posseïdor de la clau privada corresponent. La criptografia de corba el·líptica (Ed25519, ECDSA) no disposa d'una permutació trampa equivalent, de manera que l'ús de claus RSA no és una preferència sinó un requisit algorítmic de l'esquema.

En establir una connexió, els peers intercanvien les seves claus públiques RSA via el protocol /myapp/keyex/1.0.0 i les emmagatzemen en un *KeyRegistry* local, que es reconstrueix a partir de les *RingVersions* persistides en cada arrencada, sense necessitat de reconexió.

4.4 Signatures d'anell RST

S'ha implementat l'esquema RST [7] sobre claus RSA de 2048 bits amb un domini comú $b = 2208$ bits ($2048 + 160$), utilitzant AES-256-CTR com a funció simètrica E_k de la cadena de combinació, on la clau k és el SHA-256 del missatge signat (el commit, serialitzat canònicament excloent el camp *Signature*). Els 160 bits addicionals fan que la probabilitat que la permutació estesa $g_i(x)$ no sigui aplicable per a un valor aleatori sigui d'aproximadament 2^{-160} , valor negligible que garanteix la bijectivitat necessària per a l'esquema.

Les dues operacions centrals són *RingSign*, que produeix una signatura per a un missatge a partir de les claus públiques de tots els membres de l'anell i la clau privada del signant—generant valors aleatoris per a la resta de membres i resolent l'equació de tancament del circuit per a la posició

del signant sense revelar-la–, i `RingVerify`, que reconstrueix la cadena de combinació $C_{k,v}(y_1, \dots, y_r)$ i comprova que el resultat coincideix amb el valor d'ancoratge v . Les funcions `SignCommit` i `VerifyCommit` encapsulen la serialització, la selecció de la *RingVersion* corresponent a l'Epoch del commit, i la conversió de claus entre els formats `libp2p` i `RSA` estàndard.

La política de seguretat aplicada és de **rebuig estricte**: qualsevol commit sense signatura, amb signatura invàlida o sense *RingVersion* disponible per al seu Epoch és rebutjat pel node receptor durant la sincronització, juntament amb tots els seus descendents al DAG (invalidació en cascada). Aquesta verificació es distribueix entre tots els nodes membres, de manera que la manipulació local d'un commit o d'una signatura és detectable en sincronitzar.

4.5 Propagació automàtica i deniability plausible

El sistema implementa un mecanisme de *gossip flooding*. En crear un commit, el node executa `propagateCommit`: l'afegeix al seu `SeenSet`, i per a cada membre connectat llança en paral·lel `pushBlobToPeer (/myapp/blobpush/1.0.0)` seguit de `sendCommitToMembers (/myapp/commit/1.0.0)`. El *handler* passiu rep el commit, consulta el `SeenSet`, verifica la signatura `RST`, integra el commit al DAG local i el reenvia a la resta de membres connectats, excloent el remitent.

Atès que tots els membres connectats reben el contingut de manera simultània i paral·lela, la possessió d'un *blob* no implica autoria: aquesta estratègia proporciona **deniability plausible** a nivell de xarxa, ja que un observador extern no pot determinar quin node ha generat el contingut únicament a partir de qui el posseeix. Identificar l'autor requeriria un atac de correlació temporal actiu, que queda fora de l'abast d'aquest prototip.

4.6 Govern distribuït: propostes i quòrum bizantí

A diferència de les signatures d'anell dels commits –intencionadament anònimes–, el sistema de *proposals* utilitza signatures `RSA-PSS` [18] **atribuïbles** sobre les mateixes claus `RSA-2048` de cada membre, ja que la identitat del votant ha de ser verificable per comptar el quòrum. S'han implementat les funcions `SignProposal`, `VerifyProposalSignature` i `HashProposalCore`, que operen sobre el `SHA-256` del JSON canònic del `ProposalCore`.

Per crear una proposta (`create proposal <projecte> <índexs>`), un membre construeix una `ProposedVersion` amb `Epoch = ValidVersion.Epoch + 1` i la nova llista de membres ordenada alfabèticament per `PeerID`; calcula el llindar $\text{Threshold} = \lfloor 2n/3 \rfloor$, on n és el nombre de membres de la *RingVersion* vigent –aplicant directament el resultat de Lamport, Shostak i Pease [13] sobre tolerància a fallades bizantines–; signa la proposta com a primer vot; la persisteix en disc (`PROPOSAL-<projecte>_<epoch>_<hash>.json`), amb el nom derivat determinísticament del *hash* del

`ProposalCore`, garantint que dos nodes que creïn independentment la mateixa proposta produeixin el mateix fitxer); i finalment realitza un **sync global** previ a la propagació –pas de seguretat necessari, ja que acceptar la proposta sense sincronitzar primer invalidaria en cascada qualsevol commit de l'Epoch actual encara no propagat.

Cada membre elegible que rep la proposta pot votar (`vote <hash>`), afegint la seva signatura `RSA-PSS` (una signatura `RSA` no determinista amb resistència a falsificació) en mode **append-only** (les signatures mai s'eliminen del fitxer). En rebre una proposta per gossip, cada node recalcula el `Threshold` des de zero (no es confia en el valor del remitent), normalitza l'ordre dels membres, verifica cada signatura individualment –descartant silenciosament les invàlides o duplicades– i fusiona les noves firmes a la còpia local. Quan el nombre de signatures vàlides supera el `Threshold`, el node aplica automàticament la `ProposedVersion` com a nou *RingVersion* actiu, actualitza la llista de membres del projecte i persisteix l'estat. El fitxer de la proposta no s'elimina en aprovar-se, i queda com a registre històric. Les propostes que no assoleixen el quòrum romanen persistides en disc indefinidament i poden acumular nous vots en qualsevol moment posterior; no existeix mecanisme de caducitat. Si una proposta concurren per al mateix Epoch ja ha estat aprovada, les signatures addicionals es continuen processant però l'aplicació de la *RingVersion* no té efecte (*first-one-wins*): el fitxer roman en disc com a registre històric però queda inert.

La regla de desempat per a propostes concurrents del mateix Epoch és *first-one-wins* per node: la primera proposta que assoleix el quòrum localment ja satisfà per definició la condició de tolerància a fallades bizantines, independentment de l'existència de propostes concurrents, de manera que determinar quina acumula més suport en temps real no és necessari.

4.7 Persistència de l'estat

L'estat complet d'un node –projectes, membres, *RingVersions* i el DAG de commits de cada fitxer– es persisteix en format JSON a `/data/state.json`, escrit de manera atòmica (a un fitxer temporal, seguit d'un `rename`) per evitar corrupció en cas d'aturada abrupta. Les *proposals* pendents de vot es persisteixen com a fitxers independents, `PROPOSAL-<projecte>_<epoch>_<hash>.json`, amb nom derivat determinísticament del *hash* del `ProposalCore` (Secció 4.6).

En carregar l'estat a l'arrencada, `loadState` realitza una validació en dos passos: primer fa un *unmarshalling* (deserialització de JSON a estructures de dades en memòria) a una estructura intermèdia que preserva el camp `Signature` com a JSON cru, per detectar signatures amb codificació `base64` invàlida que el descodificador estàndard de Go ignoraria silenciosament (deixant `Signature` a `nil` sense error). Si es detecten signatures corruptes, el node ho notifica explícitament i avisa que aquests commits, tot i restar al DAG local, seran rebutjats per altres peers en sincronitzar –coherent amb la política de rebuig estricte de la secció 4.

4.8 Model d'amenaça i propietats de seguretat

Aquesta secció sintetitza, a partir dels mecanismes descrits, quines garanties de seguretat ofereix el sistema, sota quines suposicions, i quins atacs queden explícitament fora del seu abast.

Adversari considerat. Es distingeix entre adversaris *externs* (no membres d'un projecte, descartats per la comprovació `HasMember` a tots els protocols) i adversaris *interns* (membres legítims que actuen de mala fe). Un adversari pot controlar diversos nodes simultàniament (col·lusió); per a la tolerància bizantina del mecanisme de govern (Secció 4.6) s'assumeix que el nombre de membres deshonestos és inferior a $n/3$, el complement del quòrum $\lfloor 2n/3 \rfloor$.

Suposicions de confiança. (a) almenys $\lfloor 2n/3 \rfloor$ dels membres del *ring* vigent actuen honestament; (b) cada membre custodia la seva clau privada RSA, i la seva filtració compromet únicament la identitat criptogràfica d'aquell node –la resolució requereix comunicació fora de banda, típicament una *proposal* d'expulsió iniciada pel propi afectat–; (c) el descobriment mDNS[17] opera sobre una xarxa local de confiança, sense un adversari capaç d'injectar-se a la xarxa de transport; i (d) la incorporació a un projecte encara desconegut es basa en confiança del tipus *trust-on-first-use*: un node adopta la *RingVersion* i la llista de membres rebudes del primer peer amb qui sincronitza aquell projecte.

Propietats garantides. El disseny ofereix les següents garanties:

- *Anonimat incondicional intern:* gràcies a RST, cap membre –ni cap observador amb temps i recursos computacionals il·limitats– pot determinar quina clau de l'anell ha generat una signatura concreta.
- *Pertinença i integritat verificables:* tot commit ha de portar una signatura RST vàlida sobre la *RingVersion* del seu Epoch; els commits sense signatura o amb signatura invàlida es rebutgen, junt amb tota la seva cadena de descendents (invalidació en cascada).
- *Aïllament de la composició del grup:* Members i RingVersions d'un projecte ja conegut només es poden modificar mitjançant `applyQuorum`, és a dir, amb el consens $\lfloor 2n/3 \rfloor$ d'una *proposal*. Cap missatge de `sync` ni de *gossip* pot alterar-los directament: la fusió de l'estat (`mergeSyncMessage`) només afegeix fitxers i commits a un projecte existent, mai en redefeix els membres. Així, la injecció de membres falsos en un projecte ja establert és estructuralment impossible, no només detectable a posteriori.
- *Control unilateral de la difusió:* la recepció de trànsit d'un projecte (per `sync` o per *gossip*) la decideix sempre el node emissor honest segons la seva pròpia còpia de Members (`buildSyncMessage`, `sendCommitToMembers`, `propagateProposal`); un node no pot auto-subscriure's al trànsit d'un projecte editant únicament el seu propi estat local.
- *Tolerància a fallades bizantines en la governança:* el llinard $\lfloor 2n/3 \rfloor$ [13] garanteix que no es puguin apro-

var dues *RingVersions* contradictòries per al mateix Epoch, ni tan sols en presència de particions de xarxa.

Explícitament fora de l'abast. El sistema no contempla: (i) atacs a la capa de xarxa –interceptació, repetició o denegació de servei (MITM, DoS) sobre el trànsit P2P, correlació temporal de trànsit per desanonimitzar autors, o manipulació de l'estat rebut per un node durant la seva primera incorporació a un projecte (confiança *trust-on-first-use*) – (ii) atacs d'enginyeria social per obtenir claus privades (*phishing*); i (iii) desanonimització per **estilometria**, és a dir, marcadors d'estil personal –estil de codi, redacció dels missatges de commit– que el propi contingut pugui revelar, i que cap mecanisme criptogràfic pot neutralitzar.

5 RESULTATS I VALIDACIÓ

El prototip s'ha validat en un entorn de tres contenidors Docker (tres nodes peer), construïts a partir d'una mateixa imatge que conté el binari Go compilat, connectats mitjançant una xarxa interna de Docker que simula un entorn LAN amb descobriment automàtic via mDNS[17].

El procediment de validació ha cobert: (1) arrencada simultània dels nodes i generació d'identificadors únics; (2) descobriment automàtic de peers dins la xarxa interna; (3) creació de projectes i fitxers via CLI, amb generació i signatura RST dels commits resultants; (4) sincronització de metadades i propagació per gossip, comprovant que els commits arriben a tots els membres connectats; (5) transferència i verificació d'integritat de *blobs* mitjançant hash; i (6) consistència eventual davant desconnexions i reconexions de nodes.

S'ha validat el comportament de seguretat davant commits manipulats mitjançant el flux següent: s'extreu el fitxer `/data/state.json` d'un node via `docker cp`, es corromp manualment el camp `signature` d'un commit, i es reintrodueix l'estat. Per garantir que el procés recarregui l'estat des del disc cal fer `docker compose down/up`, que reinicia el binari; `stop/start` mantindria el procés en memòria i ignoraria el canvi. En sincronitzar, el node receptor emet un missatge `[security] REJECTED` per al commit amb signatura invàlida, seguit d'un `[security] rejected ... cascade` per a cada descendent al DAG. El node afectat apareix al `project overview` amb el fitxer sense commits vàlids. El node local no autoverifica les signatures pròpies en carregar l'estat, però `loadState` detecta base64 no decodificable i avisa que aquests commits seran rebutjats per altres peers en sincronitzar. La sortida completa d'aquesta prova es recull a la Figura 3 de l'apèndix.

També s'ha validat el cicle complet del sistema de *proposals*. El node 1 executa `create proposal` per proposar una nova composició del grup (epoch 2 amb 2 membres, excloent el node 3) i vota implícitament com a primer signant. Els nodes 2 i 3 reben la proposta per gossip i el sistema els habilita temporalment per emetre el seu vot via `vote <hash>`. En emetre el node 2 el seu vot, s'assoleix el quòrum $\lfloor 2n/3 \rfloor = 2$ signatures sobre 3 membres i la nova *RingVersion* s'activa automàticament a tots els nodes. El node 3 rep la notificació d'activació però perd l'accés al projecte: en no formar part de l'Epoch 2, cap sincronització futura li enviarà trànsit del projecte i els seus commits

serien rebutjats per signatura d'epoch invàlid. El flux complet de missatges d'aquest cicle es recull a la Figura 4 de l'apèndix.

Els resultats confirmen que el prototip compleix les propietats dissenyades: anonimat incondicional dels autors de commits, verificabilitat de pertinença al grup, rebuig estricte de contribucions no signades o invàlides, i govern descentralitzat de la composició del grup tolerant a particions de xarxa.

6 CONCLUSIONS

Els objectius generals i específics del treball s'han assolit. S'ha dissenyat i implementat un prototip funcional d'un sistema distribuït de control de versions que permet contribucions anònimes verificables dins d'un conjunt definit de membres. L'esquema de signatura d'anell RST garanteix que qualsevol commit prové d'un membre legítim del projecte sense revelar quina identitat concreta l'ha generat, i el mecanisme de propagació de *blobs* descrit a la Secció 4.5 reforça aquesta propietat amb deniabilitat plausible a nivell de xarxa. La gestió descentralitzada de claus s'ha resolt mitjançant l'estructura de *RingVersions* i el *KeyRegistry*, i el sistema de *proposals* amb quòrum materialitza la validació col·lectiva dels canvis en la composició del grup, sense autoritat central.

6.1 El límit de la criptografia davant la governança

La part conceptualment més rellevant d'aquest treball no és únicament la implementada, sinó també la investigada i descartada durant el disseny. Per resoldre el problema de la governança –identificar un membre maliciós dins d'un sistema anònim– es va explorar la incorporació de signatures d'anell vinculables (LSAG [8]), un esquema que permetria reconstruir la identitat d'un membre a partir d'un conjunt de signatures pròpies. Aquesta investigació va portar a una conclusió clara: intentar resoldre la governança des de dins de l'anonimat implica erosionar-lo necessàriament. Qualsevol mecanisme que permeti identificar un autor en determinades circumstàncies debilita estructuralment les garanties d'anonimat per a tots els casos. La criptografia no pot garantir simultàniament anonimat complet i responsabilitat individual; triar l'una implica cedir en l'altra.

Per aquest motiu, el sistema de *proposals* implementat gestiona la composició del cercle de confiança criptogràfica, però no resol qui ha de ser-ne expulsat ni per quins motius: aquesta decisió requereix un consens construït fora del sistema anònim, en l'espai de la confiança humana i la comunicació externa entre participants. La revocació, doncs, és un procés social i de protocol –excloure mitjançant una proposta–, no un procés criptogràfic.

6.2 Línies futures

Aquest treball s'ha centrat principalment en la capa criptogràfica. La resistència a atacs de xarxa, correlació temporal o anàlisi de tràfic és igualment important per al funcionament correcte d'un sistema d'aquesta envergadura i constitueix una línia de treball futur natural, així com l'extensió

del descobriment de peers més enllà de la xarxa local (nodes de *bootstrap*, DHT - Distributed Hash Table).

6.3 Reflexió final

La conclusió central d'aquest treball és que, en un sistema distribuït anònim, els problemes de governança de grup han de sortir necessàriament fora de l'anonimat: el *qui* –a qui es confia, a qui s'accepta, a qui s'expulsa– depèn de relacions de confiança externes que cap protocol criptogràfic pot substituir. El treball conclou amb un prototip tècnicament funcional que compleix els objectius proposats i que, alhora, posa de manifest on acaba la competència de la criptografia i on comença la de la confiança humana.

ÚS DE LA IA GENERATIVA

Durant el desenvolupament d'aquest treball s'han utilitzat dues eines d'intel·ligència artificial generativa. ChatGPT s'ha emprat com a suport auxiliar general, principalment en les fases inicials del projecte: com a eina d'exploració de contingut acadèmic per identificar àrees de recerca rellevants, com a corrector de text, i com a suport en l'estructuració preliminar de continguts. Claude s'ha utilitzat de manera més intensiva, amb tres rols diferenciats: suport en la redacció i estructuració de l'informe, assistència en la implementació iterativa del codi en Go, i traducció a codi de la lògica criptogràfica descrita en fonts acadèmiques.

Pel que fa a la **redacció**, l'autor ha elaborat els continguts tècnics i ha definit les decisions de disseny; Claude ha actuat com a eina de suport lingüístic i estructural, ajudant a expressar en llenguatge acadèmic el disseny i el codi implementat, i reestructurant seccions a partir de materials i indicacions prèviament elaborats per l'autor.

Pel que fa a la **implementació en Go**, llenguatge sense experiència prèvia per part de l'autor, les parts més estàndard (registres, REPL, serialització JSON, protocols *libp2p*) s'han desenvolupat de manera iterativa, amb l'autor escrivint codi i corregint errors amb suport d'IA; per a les parts de major complexitat, Claude ha implementat blocs de codi a partir d'especificacions definides per l'autor, sota la seva supervisió.

Pel que fa a la **criptografia** –l'esquema RST, la funció de combinació, i el mecanisme de *proposals* amb RSA-PSS–, el disseny es basa en referències acadèmiques seleccionades per l'autor, i la traducció a Go s'ha realitzat amb una dependència substancial de Claude, aportant l'autor els documents acadèmics originals com a referència canònica i supervisant l'adherència de la implementació a l'esquema original. L'autor comprèn el funcionament conceptual de la criptografia, però reconeix que la implementació concreta en Go –incloent l'ús de les llibreries criptogràfiques corresponents– no hauria estat possible sense assistència d'IA.

El disseny conceptual, les decisions arquitectòniques, l'elecció de les referències acadèmiques i la supervisió de totes les parts implementades són responsabilitat exclusiva de l'autor, que és capaç d'explicar el funcionament del sistema en la seva totalitat i de justificar totes les decisions de disseny. La IA no s'ha utilitzat com a font acadèmica ni ha

substituït la revisió bibliogràfica formal; tot el codi generat ha estat revisat, provat i integrat per l'autor, i en cas de discrepància s'ha prioritzat la fidelitat a la implementació acadèmica de referència per sobre de les alternatives proposades per la IA.

AGRAÏMENTS

L'autor vol agrair a la tutora del treball, Cristina Pérez Solà, l'orientació i el seguiment proporcionats al llarg de totes les fases del projecte.

REFERÈNCIES

- [1] O. Johansson and D. Persson, "Workflows and distributed version control," in *Proceedings of the Conference on Open Source Systems*, 2013.
- [2] C. Brindescu, M. Codoban, S. Shmarkatiuk, and D. Dig, "How do centralized and distributed version control systems impact software changes?" in *Proc. 36th International Conference on Software Engineering (ICSE)*, 2014.
- [3] D. B. Terry, M. M. Theimer, K. Petersen, A. J. Demers, M. J. Spreitzer, and C. H. Hauser, "Managing update conflicts in Bayou, a weakly connected replicated storage system," in *Proc. 15th ACM Symposium on Operating Systems Principles (SOSP)*, 1995.
- [4] S. Haber and W. S. Stornetta, "How to time-stamp a digital document I," *Journal of Cryptology*, 1991.
- [5] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology – CRYPTO '87*, LNCS, 1987.
- [6] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, 1976.
- [7] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *Advances in Cryptology – ASIACRYPT 2001*, LNCS, Springer, 2001.
- [8] J. K. Liu, V. K. Wei, and D. S. Wong, "Linkable spontaneous anonymous group signature for ad hoc groups," in *Information Security and Privacy – ACISP 2004*, LNCS, Springer, 2004.
- [9] D. Chaum and E. van Heyst, "Group signatures," in *Advances in Cryptology – EUROCRYPT '91*, LNCS, Springer, 1991.
- [10] C. Komlo, "Threshold signatures," *IEEE Security & Privacy*, Nov./Dec. 2024.
- [11] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," *Journal of Cryptology*, 2007.
- [12] W. W. Royce, "Managing the development of large software systems," in *Proc. IEEE WESCON*, 1970.
- [13] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, 1982.
- [14] J. Kulatunga, "Gitmask: Contribute code anonymously," 2018. [Online]. Available: <https://github.com/AnalogJ/gitmask>. [Accessed: June 2026].
- [15] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.
- [16] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," *Journal of Cryptographic Engineering*, vol. 2, no. 1, pp. 77–89, 2012.
- [17] S. Cheshire and M. Krochmal, "Multicast DNS," IETF RFC 6762, February 2013.
- [18] K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2," IETF RFC 8017, November 2016.

APÈNDIX

A.1 Comandes principals de la CLI

Xarxa i peers

- `help`: mostra la llista de comandes disponibles.
- `id`: imprimeix l'identificador únic del peer (`PeerID`, derivat de la identitat `Ed25519`) i les adreces de xarxa associades.
- `peers`: mostra la llista de peers descoberts localment i el seu estat (`known/connected`).
- `connect <índex>`: estableix una connexió amb un peer conegut, intercanvia les claus públiques RSA (`/myapp/keyex/1.0.0`) i desencadena la sincronització de metadades.
- `sync <índex>`: inicia manualment una sincronització amb un peer ja connectat.
- `pull`: recupera els *blobs* absents dels projectes de l'usuari.

Projectes i fitxers

- `project create <nom> <i> [i...]`: crea un nou projecte definint els membres per índex (–1 inclou el propi node), construint la *RingVersion* de l'Epoch 1 amb les claus RSA de cada membre.
- `project list`: mostra els projectes dels quals el peer local és membre.
- `project overview <nom>`: mostra una visió detallada d'un projecte (fitxers, commits, *blobs*).
- `file create <projecte> <fitxer> <missatge>`: afegeix un nou fitxer a un projecte i crea el primer commit, signat amb RST sobre l'Epoch actiu, amb propagació automàtica per gossip.

- `version create <projecte>`
`<fitxer.projecte> <fitxer.disc>`
`<parentID, ...> <missatge>`: crea un nou commit sobre un fitxer existent, permetent seleccionar un o diversos commits pare (fusions de branques); aplica l'Epoch de la *RingVersion* activa, signa amb RST i propaga el *blob* i les metadades per gossip.
- `file log <projecte> <fitxer>`: mostra el DAG de commits d'un fitxer en format ASCII, amb l'Epoch, la data i el missatge de cada node.

Governança

- `create proposal <projecte>`
`<idx1,idx2, ...>`: proposa una nova *RingVersion*, resolent els índexs de peers a membres amb les seves claus públiques RSA; el node creador vota implícitament amb la primera signatura RSA-PSS i fa un *sync* global abans de propagar.
- `vote [<hash>]`: sense arguments, llista les propostes pendents de vot amb el nombre de signatures acumulades; amb un *hash*, afegeix la signatura RSA-PSS del node local, persisteix la proposta, comprova el quòrum $\lfloor 2n/3 \rfloor$ i la propaga.

Fig. 4: Cicle complet d'una proposta d'expulsió: creació al node 1, recepció per gossip als nodes 2 i 3, i aprovació en assolir el quòrum amb 2 signatures. Node 3 rep la notificació però queda exclòs de l'Epoch 2.

A.2 Sortides de consola de les proves

```
> connect 0
Connecting to 12D3KooWRXsW... Connected!
[keyex] stored public key for 12D3KooWRXsW...
[sync] received state from 12D3KooWRXsW...

[security] REJECTED 56a82690... ("Initial commit"):
  invalid signature -- unmarshaling ring signature:
  invalid character '0' in string escape code

[security] rejected commit 0bd039d0... ("Second commit"):
  cascade from invalid parent 56a82690...

[sync] rejected commit 56a82690...
[sync] rejected commit 0bd039d0... (cascade)
[sync] new project "myrepo" added
[pull] all blobs present locally -- nothing to fetch

> project overview myrepo
project : myrepo  members : 3  files : 1
         app.js   (no commits)
```

Fig. 3: Rebuig d'un commit amb signatura manipulada i invalidació en cascada del seu descendent. El fitxer queda sense commits vàlids al node receptor.

```
(nodes 2 i 3, per gossip)
[proposal] New proposal for project 'myrepo'
           has arrived (epoch 2,
           hash c5603abea57ee90d).
           You have been temporarily enabled
           to use the 'vote' command.

(node 2)
> vote c5603abea57ee90d
[vote] Vote cast for proposal c5603abea57ee90d
       (epoch 2 in project 'myrepo').
       Signatures: 2/2
[proposal] Proposal c5603abea57ee90d for
           'myrepo' has reached quorum.
           Epoch 2 is now active. Ring updated.
```