UNIVERSITAT D'ANDORRA

Programa de doctorat de la Universitat d'Andorra

Credencials anònimes enllaçables: com aplicar la identitat autogestionada a comptadors d'ús en un entorn hostil amb tecnologia blockchain (Tesi Doctoral del programa de doctorat)

Francesc Garcia Grau

Direcció: Dr. Jordi Herrera Joancomartí i Dr. Aleix Dorca Josa
Identificador: TD-127-108590/202503
Data de defensa: 15 de juliol de 2025

TESI DOCTORAL

Credencials anònimes enllaçables: com aplicar la identitat autogestionada a comptadors d'ùs en un entorn hostil amb tecnologia blockchain

Francesc Garcia Grau

2025

TESI DOCTORAL

Credencials anònimes enllaçables: com aplicar la identitat autogestionada a comptadors d'ùs en un entorn hostil amb tecnologia blockchain

DOCTORAL THESIS

Linkable anonymous credencials: how to apply self sovereign identity to electronic voting with Blockchain and Smart Contracts

Francesc Garcia Grau

2025

PROGRAMA DE DOCTORAT

Dirigida per:

Jordi Herrera Joancomartí

Departament d'Enginyeria de la Informació i les Comunicacions
Universitat Autònoma de Barcelona
CYBERCAT – Center for Cybersecurity Research of Catalonia

Aleix Dorca Josa

Departament de Sistemes informàtics
Universitat d'Andorra
GRET – Grup de recerca en tecnologia

Memòria presentada per optar al títol de doctor/a per la Universitat d'Andorra

Per qui em va engrescar a l'inici.
M'ha acompanyat durant tot el camí.
I sense l'ajuda de qui no ho hauria aconseguit.
Gràcies, Mònica.

Agraïments

En primer lloc, vull expressar el meu més profund agraïment als meus pares, per la seva educació, dedicació i suport incondicional. Sense la seva guia i el seu amor constant, no hauria estat capaç de superar les dificultats que he trobat al llarg d'aquest camí. Ells han estat la base sobre la qual he pogut construir els meus somnis i metes.

Als meus directors de tesi, agraeixo sincerament l'ajuda i el suport que m'han brindat durant tot aquest procés. La vostra paciència, consells valuosos i orientació han estat crucials per a la meva recerca. Sé que no ha estat un camí fàcil, però la vostra comprensió i dedicació han fet que aquest projecte fos possible.

Vull també agrair a l'empresa i als meus companys de feina per la seva comprensió i suport al llarg d'aquest període. Gràcies per permetre'm gaudir de la flexibilitat de temps que necessitava per poder compaginar la meva tasca professional amb la meva dedicació a la tesi. La vostra col·laboració ha estat essencial perquè pogués mantenir aquest equilibri.

Finalment, un agraïment molt especial als meus éssers estimats, per la seva paciència i recolzament durant aquest llarg procés. Gràcies per comprendre la meva dedicació, malgrat la meva absència, i per estar sempre al meu costat, donant-me forces per seguir endavant.

Índex

Índex de figures

Índex de taules

Abreviatures

$\mathcal{AP}$ Attribute Provider. vi, ix, 47, 50, 51, 57, 59–62, 66–68, 70, 72–74, 81–83, 85

$\mathcal{AV}$ Atribut Verificable. ix, 16, 17, 19, 43, 47

$\mathcal{CP}$ Certificate Provider. 36–39, 50–52, 70

$\mathcal{IV}$ Identity Validator. 36, 37, 50, 70

$\mathcal{SP}$ Service Provider. vi, ix, 36–39, 47, 50–52, 54–57, 59–70, 72, 75, 76, 86

$\mathcal{U}$ Usuari. vi, ix, 36–39, 47, 50–52, 55, 59–70, 73–76

$id_{\mathcal{U}}$ identificador únic d'usuari. vi, vii, 53, 55–57, 61, 67–72, 77, 80–83, 85, 86, 109

API Application Programming Interface. 70, 71, 77

BDHP Bilinear computational Diffie–Hellman problem. 11

BFT Byzantine Fault Tolerance. 32

BGR Block Generation Rate. ix, 89, 91–93

BTC Bitcoin. 28, 31

CAS Content Addressable Storage. 33

CDHP Computational Diffie–Hellman. 11, 68

CPU Central Process Unit. 31, 32

creds Credencials. 13

DA Dades d'Autenticació. 20, 21

DCDHP Divisible computation Diffie–Hellman problem. 11, 68

DID Distributed Identifiers. v, ix, 20–22, 24, 25

DL Distributed Ledger. 27, 28

DLP Discret Logarithm Problem. 11, 55, 68, 69

DLT Distributed Ledger Technology. 27, 33

DP Dades Privades. 13, 20, 21

EC Entitats de Confiança. 16, 17

ECC Elliptic Curve Criptography. 59

ECDSA Elliptic Curve Digital Signature Algorithm. 36

EIP Ethereum Improvement Proposals. 80, 81, 97, 98

ETH Ether. 34

EVM Ethereum Virtual Machine. 34, 35

GPU Graphics Process Unit. 31, 32

IBC Identity-Based Cryptography. 10

IDM Identity Manager. v, ix, 14–16

IKE Internet Key Exchange. 10

Inv-CDHP Inverse computational Diffie–Hellman problem. 11, 68

IoI Items of Interest. 5

IPFS InterPlanetary File System. 33

JSON JavaScript Object Notation. 22, 70–76, 82, 84

LGB Limit Gas Block. ix, xi, 89–91, 93

NI-Schnorr Non Interactive Schnorr Zero Knowledge Proof. 36, 37, 39, 52

NI-ZKP Non Interactive Zero Knowledge Proof. 8, 10, 54, 57

NIST National Institute of Standards and Technology. 28, 35

OAuth Open Authorisation. 14–16

PKI Public Key Infrasctructure. 10

PoA Prof-of-Authority. 31, 32

PoS Prof-of-Stake. 31, 32

PoW Prof-of-Work. 31, 32

Privacy-ABC Privacy-preserving Attribute Base Credencials. 17

REST Representational State Transfer. 70, 71, 77

SAML Security Assertion Markup Language. 14–16

SC Smart Contract. vi, xvii, xix, 1, 2, 33, 34, 43, 45–47, 49, 56–58, 60, 61, 64–66, 69, 70, 80, 81, 84, 86–89

SHA Secure Hash Algorithm. 28

SHA3-256 Secure Hash Algorithm versió 3 de 256bits. 28, 35

SI Sistemes Informàtics. 13–16, 18, 20, 21, 88

Sqr-CDHP Square computational Diffie–Hellman problem. 11

SSI Self Sovereign Identity. v, ix, xvii, xix, 1, 2, 17–20, 24, 25, 47

SSO Single Sign On. 15

TI Tecnologies de la Informació. 45

TpS Transaccions per Segon. ix, 88–93

URN Universal Resources Notation. 22

VC Verifiable Claim. v, ix, 23, 24

W3C World Wide Web Consortium. 5, 19, 21

YAML Yet Another Markup Language. 89

ZKP Zero Knowledge Proof. 6–8, 47, 55, 56, 59–61, 63, 75, 76, 87, 109

ZSS Zhang, Safavi, and Susilo. v, 11, 36–38, 50, 51, 53, 61, 62, 67, 68

Resum

Aquesta tesi explora l'aplicació de les credencials anònimes enllaçables com a part d'una Self Sovereign Identity (SSI) per a protegir la privadesa i la seguretat dels usuaris, en particular, en entorns hostils. El treball se centra en el desenvolupament d'un sistema capaç de garantir la verificació segura de la identitat, especialment per a comptadors d'ús en entorns vulnerables, utilitzant tecnologies emergents com la tecnologia blockchain.

Les SSI permeten als usuaris controlar completament les seves credencials, evitant la dependència d'entitats centralitzades, i les credencials anònimes enllaçables permeten validar la identitat sense exposar dades personals sensibles. Aquest model proposa una solució robusta per a minimitzar els riscos associats a l'emmagatzematge i la transmissió de dades personals en línia, mantenint un alt nivell de privadesa per als usuaris.

Mitjançant l'ús de la tecnologia blockchain, es crea un registre immutable i transparent que garanteix la seguretat de les credencials, mentre que els Smart Contracts (SCs) automatitzen els processos de validació, permetent interaccions segures i eficients sense necessitat de confiança en intermediaris. Aquesta tesi proposa una arquitectura tecnològica que integra aquestes eines per a dissenyar un sistema segur i privat, adaptat a les necessitats específiques dels entorns hostils en què la protecció de la identitat és crucial.

Abstract

This thesis explores the application of linkable anonymous credentials as part of a SSI system to protect user privacy and security, particularly in hostile environments. The work focuses on developing a system capable of ensuring secure identity verification, especially for public accountants operating in vulnerable contexts, using emerging technologies such as blockchain technology.

SSIs allow users to fully control their credentials, eliminating the need for centralized entities, while linkable anonymous credentials enable identity validation without exposing sensitive personal data. This model proposes a robust solution to minimize the risks associated with storing and transmitting personal data online, while maintaining a high level of user privacy.

By leveraging blockchain technology, an immutable and transparent ledger is created, ensuring the security of credentials, while SCs automate validation processes, allowing secure and efficient interactions without the need for trust in intermediaries. This thesis proposes a technological framework that integrates these tools to design a secure and private system, tailored to the specific needs of hostile environments where identity protection is critical.

Capítol 1

Introducció

En l'era digital actual, la seguretat i la privadesa de les dades personals són qüestions de creixent importància. L'auge de la identitat digital ha generat un debat intens sobre com protegir la informació personal i garantir que només les parts autoritzades puguin accedir-hi. A la vegada, la necessitat de mantenir l'anonimat, especialment en entorns digitals on els usuaris poden ser subjectes a vigilància o persecució, s'ha convertit en un repte fonamental.

Aquest treball explora la manera d'implementar un sistema de credencials anònimes enllaçables com a part d'un sistema Self Sovereign Identity (SSI). Aquest tipus d'identitat permet als individus gestionar les seves pròpies credencials sense dependre d'una entitat centralitzada, garantint un control total sobre la seva informació personal. Aquesta solució es veu especialment rellevant en entorns hostils, com el cas dels comptadors d'ùs, on la seguretat i la privadesa són vitals.

Per aconseguir-ho, la tesi proposa l'ús de la tecnologia blockchain i mes concretament els Smart Contract (SC) com a eines per garantir la seguretat, la transparència i la verificabilitat de les credencials sense comprometre l'anonimat dels usuaris. La tecnologia blockchain, amb la seva naturalesa descentralitzada i immutable, proporciona una base sòlida per emmagatzemar i verificar informació de manera segura, mentre que els SCs permeten automatitzar i executar processos de manera autònoma i transparent.

Aquest estudi s'endinsa en la manera com aquestes tecnologies poden ser integrades per crear un sistema de verificació d'identitats anònimes, segures i autèntiques, que pugui ser utilitzat per comptadors d'ùs, a la vegada que preserva la privadesa dels individus i minimitza els riscos derivats de l'ús de dades personals en entorns vulnerables. En aquest context, un comptador d'ús fa referència al registre de quantes vegades s'ha utilitzat una credencial específica per obtenir el dret o accés associat a aquesta credencial.

A través d'una anàlisi detallada de la teoria darrere de les identitats autogestionades, així com de les aplicacions pràctiques de Blockchain i SCs, es pretén demostrar com aquestes innovacions tecnològiques poden transformar el món de la identitat digital, proporcionant solucions robustes a les amenaces en entorns hostils. En l'Apartat 1.1 s'expliquen les motivacions i el context de la recerca. Per acabar en l'Apartat 1.2 es desglossa l'estructura del document.

1.1 Motivacions i context de la recerca

La creixent digitalització de la societat comporta nombrosos avanços en la manera com interactuem i compartim informació, però també presenta reptes cada vegada més complexos pel que fa a la seguretat i la privadesa de les dades personals. A mesura que la tecnologia avança, també ho fan les amenaces a la seguretat, especialment en entorns on la protecció de la identitat és crucial, com és el cas de les administracions públiques, els serveis financers i altres sectors regulats. En aquests contextos, les credencials d'identitat tradicionals basades en models centralitzats han demostrat ser vulnerables a atacs, manipulacions o filtracions de dades sensibles, posant en perill la seguretat de la informació de la ciutadania i la confiança pública.

És essencial protegir els comptadors d'ús contra possibles modificacions il·legítimes per garantir la integritat i la fiabilitat del sistema de verificació. Qualsevol alteració no autoritzada d'aquestes dades pot comprometre la seguretat i la confiança del sistema, permetent que es manipulin les estadístiques d'ús, es generin avantatges inapropiats o es vulnerin mecanismes de control d'accés. A més, la protecció dels comptadors d'ús contribueix a preservar la privadesa dels individus, evitant que dades sensibles siguin alterades per fins malintencionats, minimitzant així els riscos associats a possibles fraus o abusos en entorns vulnerables.

En particular, els comptadors d'ùs es poden veure exposats a riscos significatius de seguretat en entorns hostils, com és el cas de països amb règims autoritaris, conflictes socials o ciberamenaces globals. La necessitat de protegir les identitats d'aquests agents dins d'aquests entorns vulnerables es fa cada cop més evident, ja que les mesures de seguretat tradicionals sovint no poden garantir un nivell suficient de privadesa ni protegir completament els usuaris.

Les SSIs ofereixen una possible solució a aquests problemes, ja que permeten que els individus gestionin les seves pròpies credencials de manera descentralitzada, donant-los més control sobre les seves dades personals. Aquest model elimina la necessitat d'un intermediarari centralitzat, reduint així els riscos associats a un possible punt únic de fallada o manipulació. A més, la possibilitat de generar credencials anònimes enllaçables permet que els usuaris validin la seva identitat de manera segura i privada, sense exposar dades personals sensibles, i facilitant la verificació sense comprometre la seguretat.

La Blockchain i els SCs ofereixen la infraestructura tecnològica per fer realitat aquest model. A través de la descentralització, la Blockchain permet registrar de manera immutable i transparent les credencials d'identitat, mentre que els SCs poden automatitzar i garantir que els processos de verificació siguin segurs i transparents sense necessitat de confiança en terceres parts.

Aquesta recerca es motiva per la necessitat de desenvolupar un sistema de credencials anònimes robust, que no només preservi la privadesa dels usuaris, sinó que també sigui aplicable en entorns on la seguretat i la fiabilitat són fonamentals. A través de la integració d'aquestes tecnologies emergents, es pretén dissenyar un sistema que no només millori la seguretat en l'administració pública, sinó que també serveixi com a referent per a altres sectors que necessiten protecció avançada de les dades personals.

En un moment en què la seguretat digital i la privadesa s'han convertit en qüestions essencials per al bon funcionament de les democràcies i els mercats globals, aquesta recerca ofereix un

model de seguretat i privadesa basat en tecnologies disruptives amb el potencial de transformar la manera en què gestionem i verifiquem les identitats en un món cada vegada més digitalitzat.

1.2 Estructura del document

Després de la introducció feta en aquest capítol, el Capítol 2 és un resum del marc teòric en el que se centra la recerca. S'estudien el mètodes d'autenticació, com han anat evolucionant en el temps i la seva relació amb l'anonimat. També es fa una introducció a la tecnologia blockchain, on s'exposen els seus punts forts i debilitats. S'aprofundeix especialment en Ethereum que és la blockchain en la que s'ha decidit dur a terme la implementació.

En el Capítol 3 s'exposa l'objectiu general així com un seguit de requisits funcionals i no funcionals que s'aniran resolent en l'estudi. Aquests són fruit de l'anàlisi detallada del marc teòric i un cop identificades les mancances existents.

El Capítol 4 exposa la metodologia emprada, seguit del Capítol 5 on es descriu el desenvolupament de la recerca i del Capítol 6 on es descriuen en detall els resultats obtinguts, resultat a resultat, amb els experiments duts a terme per validar cadascun dels resultats.

Finalment, els Capítol 7 aglutina les conclusions extretes de l'estudi i les futures línies de recerca. Per acabar, s'afegeixen les referències literàries i les cites a les contribucions fetes en aquest estudi.

L'àmbit on se centra aquesta recerca és nou, fet que explica que les referències siguin recents. Moltes tecnologies emprades sorgiren després del 2014 i els recursos es troben en línia.

Capítol 2

Marc Teòric

L'objectiu d'aquest capítol és establir el marc teòric en el qual se centra la recerca. Per començar, en l'Apartat 2.1 s'introdueix el concepte d'anonimat, per seguir en l'Apartat 2.2 s'introdueixen els avenços fets en el camp de la criptografia que han permès l'obtenció dels resultats presentats en els apartats a continuació. En l'Apartat 2.3 s'estudien diferents mètodes d'autenticació, com aquests han evolucionat en el temps i la seva relació amb l'anonimat. En l'Apartat 2.4 s'introdueix i s'estudia una implementació d'identitat autogestionada promoguda pel World Wide Web Consortium (W3C)[1]. En l'Apartat 2.5 es fa una introducció a la tecnologia blockchain. S'exposen els seus punts forts i febles. En l'Apartat 2.6 s'aprofundeix especialment en Ethereum, que és la tecnologia blockchain en la qual s'ha decidit dur a terme la implementació. I finalment, en l'Apartat 2.7 es fa una anàlisi detallada del protocol que s'ha escollit com a base de la recerca.

2.1 L'anonimat

Quan es parla d'anonimat s'ha de definir què vol dir i com s'aplica en cada context. Andreas Pfitzmann i Marit Hansen en l'article [1] defineixen formalment la terminologia utilitzada en aquest àmbit. Defineixen l'anonimat com la propietat que fa indistingible un individu determinat dins d'un conjunt d'individus que s'anomena **conjunt d'anonimat**. En el cas que ens ocupa això es tradueix en què no hi ha forma d'identificar qui fa una acció donada o envia un missatge dins el sistema.

La **no enllaçabilitat** de dos ítems d'interès (Items of Interest (IoI) p.e. individus, accions, missatge...) des del punt de vista d'un observador significa que dins del sistema (que conformen aquests ítems i possiblement altres), l'observador no pot distingir si els IoI estan relacionats o no. Per exemple, en un escenari amb més de dos individus, dos missatges enviats des del mateix **conjunt d'anonimat** no són enllaçables per un observador. Això és, que la probabilitat que aquests dos missatges hagin estat enviats pel mateix individu és prou propera a $\frac{1}{\#individus}$ on $\#individus$ és el nombre d'individus en el conjunt d'anonimat. Per tant l'existència d'anonimat implica la no enllaçabilitat. Finalment, un **pseudònim** és un identificador d'un individu altre que el seu nom real. Un individu és **pseudoanònim** si utilitza el pseudònim en lloc del seu

[1] URL: `https://www.w3.org/`.

nom real. El **pseudoanonimat** és l'ús de pseudònims com a identificadors. Notis que amb l'ús de pseudònims es pot garantir l'anonimat sense perdre l'enllaçabilitat sempre que s'utilitzi el mateix pseudònim i no sigui pública la correspondència pseudònim/identitat real. De totes maneres, si s'abusa de l'ús d'un mateix pseudònim en diferents àmbits, un observador podria inferir aquest lligam i desfer l'anonimat.

2.2 Bases de criptografia

Aquest apartat descriu els avenços més rellevants en el camp de la criptografia imprescindibles per a la solució proposada. L'Apartat 2.2.1 comença explorant les proves de coneixement nul, per seguir amb l'Apartat 2.2.2 que introdueix el aparellaments bilineals. L'Apartat 2.2.3 descriu els problemes matemàtics sobre els quals es fonamenta la criptografia utilitzada, per acabar amb els Apartats 2.2.4 i 2.2.5 que descriuen un esquema de signatura sobre aquests aparellaments bilineals i un de compromís sobre corbes el·líptiques.

2.2.1 Proves de coneixement nul

Les proves de coneixement nul van ser concebudes per primera vegada el 1985 per Shafi Goldwasser, Silvio Micali i Charles Rackoff en el seu article "The Knowledge Complexity of Interactive Proof Systems" [2]. S'anomenen Zero Knowledge Proof (ZKP), i és un protocol criptogràfic que estableix un mètode perquè una de les parts provi a l'altra part que una declaració (normalment matemàtica) és certa, sense donar cap informació més que la veracitat de la declaració. Jean-Jacques Quisquater en l'article "How to explain zero-knowledge protocols to your children" [3], mostra per primera vegada un exemple que s'ha re-interpretat de la següent manera.

Na Mònica ha descobert la paraula secreta per obrir la porta de la cova màgica d'Ali Baba. Aquesta cova té forma circular, amb una entrada en un costat, i la porta bloquejada per la paraula secreta de l'altra (Figura 2.1). En Pau, diu que li compra la paraula secreta, però vol estar segur que és la bona abans de donar-li els diners. Na Mònica, per altra banda, diu que li dirà la paraula secreta un cop hagi rebut els diners. Estableixen el següent pla perquè na Mònica pugui demostrar que sap la paraula secreta sense desvelar-la.

Primer, en Pau espera fora de la cova mentre na Mònica entra. Anomenen cada camí amb una lletra, A és el camí de l'esquerra i B el camí de la dreta (Figura 2.2). Na Mònica agafa un dels dos a l'atzar i el segueix fins a la porta. En Pau entra a la cova i anuncia a na Mònica per quin camí vol que torni, aquesta petició també es fa a l'atzar (Figura 2.3). Com na Mònica sap la paraula secreta, estigui on estigui, li és fàcil sortir pel camí designat per en Pau, obrint la porta si cal (Figura 2.4). Cal destacar que en Pau no sap per quin camí ha entrat ella.

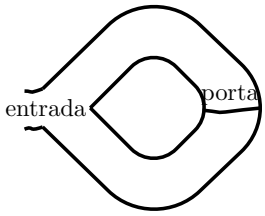

Figura 2.1: Cova d'Ali Baba.

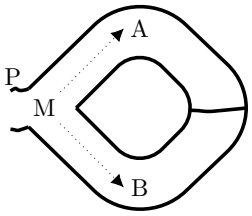

Figura 2.2: Na Mònica tria un camí a l'atzar.

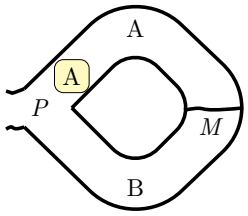

Figura 2.3: En Pol anuncia un camí de retorn.

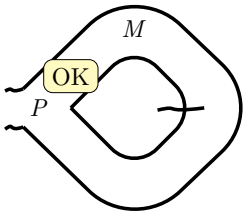

Figura 2.4: Si na Mònica és honesta retorna pel camí que en Pau anuncia.

Si més no, suposem que na Mònica no sap la paraula secreta, llavors només podria tornar pel camí per on ha entrat. Donat que en Pau ha escollit entre A o B l'atzar, na Mònica té un 50% de probabilitats d'encertar. Si repetim l'experiment un cert nombre de vegades, la probabilitat d'encertar-les totes esdevé pràcticament nul·la. Per tant, si na Mònica apareix cada cop per la sortida que en Pol designa, es pot concloure que na Mònica sap probablement la paraula secreta.

Cal notar que també seria possible demostrar que na Mònica sap la paraula secreta si ella entra per un camí i surt per l'altre, però llavors demostraria a tothom que ella sap la paraula secreta. Amb el sistema descrit anteriorment, ningú a part d'en Pau pot estar segur que ella sap la paraula secreta, ja que no poden saber si na Mònica i en Pau estan conxorxats. Un cop vist l'exemple passem a formalitzar les propietats d'una prova de coneixement nul.

Propietats de les *ZKP*

Han de complir la propietat de **correcció**. Suposem que una declaració és certa i es vol provar. Si tenim un verificador i un provador honestos (que segueixen el protocol) el protocol provarà inequívocament al verificador que la declaració és certa. Aquesta certesa implica una probabilitat d'errada molt poc significativa a la pràctica.

També ha de tenir la propietat de **robustesa**, si la declaració és falsa, no existeix un provador deshonest capaç de convèncer a un verificador honest que la declaració és certa, excepte amb probabilitat molt minsa.

I finalment, s'ha de verificar el **coneixement nul**. El verificador només obté la certesa que la declaració és certa, res més. Si el provador segueix el protocol el verificador no pot extreure cap informació nova a partir de la prova, excepte que aquesta és certa.

Les ZKP no són proves en el sentit matemàtic del terme, perquè hi ha una petita probabilitat, l'**error de robustesa** (soundness error en anglès), que un provador deshonest enganyi un verificador honest. Són proves probabilístiques i no deterministes. Encara que existeixen tècniques per reduir aquest error a valors insignificants.

Classificació de les ZKP

Les ZKP poden ser interactives o no interactives.

- **Interactives:** Cal que el verificador i el provador estiguin presents a l'hora de l'execució del protocol i solen tenir els següents passos: primer pas, el provador genera un missatge de compromís indicant que coneix el secret i el fa arribar al verificador.

  Segon pas, el verificador rep el missatge de compromís, genera un repte i el retorna al provador com a resposta al missatge de compromís. El repte sol ser aleatori i pot contenir part del missatge de compromís.

  I finalment, tercer pas, el provador retorna una resposta al verificador que pot verificar que el coneix el secret. En el càlcul de la resposta intervenen tant el repte com el secret.

  Aquest procés es repeteix diverses vegades per assegurar la verificació. Alguns exemples d'aquest tipus d'algoritme els trobem en l'algoritme d'identificació de Schnorr [4], el protocol de Chaum-Pedersen [5] i en el protocol de Cramer-Damgard-Schoenmakers [6].

- **No interactives:** En aquest cas no cal que i verificador estiguin presents durant l'execució del protocol. Per transformar un prova interactiva a una no interactiva es sol utilitzar l'heurística de Fiat-Shamir [7].

  S'anomenen Non Interactive Zero Knowledge Proof (NI-ZKP). Les proves de coneixement nul no interactives de Schonrr [8] permeten a un provador S convèncer a un verificador R que una afirmació és certa sense donar més informació que la veracitat de la prova. El protocol (S, R) ha de ser complet i sòlid.

  Tenim $\mathbb{G}$ grup d'ordre q amb generador P, i $\mathbb{Z}_q$ el camp de sencers mòdul q. El provador S coneix un valor v (clau secreta), i calcula la clau pública $y = P^v$. La informació pública és y, P i la informació privada és v.

  - S tria aleatòriament $r \in \mathbb{Z}_q$.
  - S calcula: $t = P^r$, $c = H(t)$, $s = c \cdot v + r$.
  - S envia la tupla (t, s) a R.
  - R calcula $c = H(t)$.
  - R comprova $P^s = P^{c \cdot v + r} = (P^v)^c \cdot P^r \stackrel{?}{=} y^c \cdot t$.

  La prova π és la tupla (t, s).

  Aquest esquema s'utilitza sovint sobre corbes el·líptiques com a mecanisme de verificació de la correcció de les dades compromeses durant les proves de compromís en tecnologia blockchain.

2.2.2 Aparellaments bilineals

Abans de parlar d'aparellaments bilineals, cal introduir alguns conceptes algebraics. Joseph H. Silverman, en el seu llibre "Advanced Topics in the Arithmetic of Elliptic Curves" [9] defineix

un grup additiu com una estructura algebraica que consisteix en un conjunt equipat amb una operació binària anomenada "addició"que satisfà les propietats següents:

- **Clausura**: per a qualsevol dos elements a i b del grup, la seva suma $a + b$ també pertany al grup.
- **Associativitat**: per a qualsevol tres elements a,b i c del grup, es compleix que:

$$(a + b) + c = a + (b + c)$$

- **Element neutre**: existeix un element anomenat "element neutre´ o "zero", denotat per 0, tal que per a qualsevol element a del grup:

$$a + 0 = 0 + a = a$$

- **Element invers**: per a cada element a del grup, existeix un element anomenat "invers additiu" o "oposat", denotat per $-a$, tal que:

$$a + (-a) = 0$$

Un exemple comú de grup additiu és el conjunt dels nombres enters $\mathbb{Z}$ amb l'operació d'addició habitual. Altres exemples són el conjunt dels vectors en un espai vectorial, on la suma de vectors es defineix component a component, o les corbes el·liptiques que s'utilitzen extensament en aquesta recerca.

En el context de la criptografia i els aparellaments bilineals, els grups additius cíclics són particularment importants. Aquests grups estan generats per un element anomenat "generador" i tenen una estructura cíclica, és a dir, tots els elements del grup poden ser expressats com múltiples enters de l'element generador.

Per exemple, considerem el grup additiu cíclic $\mathbb{G}_1$ generat per un element P. Si q és l'ordre del grup (el nombre d'elements en el grup), llavors cada element de $\mathbb{G}_1$ pot ser escrit com $a \cdot P$ on a és un enter entre 0 i $q - 1$.

Aquestes propietats fan que els grups additius cíclics siguin útils per a la construcció de protocols criptogràfics segurs i eficients.

S'utilitzen les definicions de l'article de Fangguo Zhang, Reihaneh Safavi-Naini i Willy Susilo, "Efficient Signature Scheme from Bilinear Pairings and Its Applications"[10] i es fan servir en aquest esquema.

Tenim $\mathbb{G}_1$ grup additiu cíclic generat per P, d'ordre $q \in \mathbb{F}_q$, i q primer. $\mathbb{G}_2$ grup multiplicatiu amb el mateix ordre q. Tenim $e : \mathbb{G}_1 \times \mathbb{G}_2$ que és una aplicació amb les següents propietats:

- **Bilinealitat**: es compleix que:

$$e(a \cdot P, b \cdot Q) = e(P, Q)^{a \cdot b}$$

per tot $P, Q \in \mathbb{G}_1$, $a, b \in \mathbb{Z}_q$

- **No degenerat**: existeixen $P, Q \in \mathbb{G}_1$ tal que $e(P, Q) \neq 1$
- **Calculabilitat**: existeix un algorisme eficient per calcular $e(P, Q)$ per tots $P, Q \in \mathbb{G}_1$

Si P és generador de $\mathbb{G}_1$, $e(P, P)$ és un generador de $\mathbb{G}_2$. Aquesta aplicació bilineal s'anomena aparellament bilineal. L'aparellament bilineal ve donat per $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.

Els aparellaments bilineals tenen diverses aplicacions importants en criptografia i altres àrees de la seguretat informàtica, com per exemple:

- **Criptografia de clau pública basada en corbes elíptiques** [11]: Els aparellaments bilineals s'utilitzen per optimitzar l'estructura de corbes elíptiques en protocols criptogràfics de clau pública. Per exemple, en la signatura d'identitat i l'intercanvi de claus.
- **Protocols de coneixement nul**: Permeten a un individu demostrar que coneix una informació secreta sense revelar-la. Això s'aconsegueix mitjançant protocols de prova d'igualtat o protocols de prova de coneixement com el protocol de Schnorr o NI-ZKP, on els aparellaments bilineals juguen un paper essencial.
- **Intercanvi de Clau Segura** [12]: En protocols com Internet Key Exchange (IKE) o Diffie-Hellman ampliat, els aparellaments bilineals poden ser utilitzats per establir una clau compartida entre dues parts de manera segura, millorant l'eficiència i la seguretat dels intercanvis.
- **Criptografia basada en identitat (Identity-Based Cryptography (IBC))**: Els aparellaments bilineals es fan servir en l'IBC, on una identitat com una adreça de correu electrònic es pot utilitzar com a clau pública, eliminant la necessitat d'infraestructures de clau pública tradicionals (Public Key Infrasctructure (PKI)). L'exemple més conegut és l'esquema de Boneh-Franklin per a signatures d'identitat [13].
- **Protocols de votació electrònica** [14]: En sistemes de votació electrònica i tecnologia blockchain, els aparellaments bilineals es poden utilitzar per garantir l'anonimat, la veracitat i la seguretat del sistema, permetent que els votants demostrin que han votat correctament sense revelar el seu vot.
- **Agregació de signatures digitals** [15]: Permeten l'agregació de signatures, és a dir, combinar múltiples signatures en una sola, mantenint la seva validesa, fet que és molt útil en aplicacions de sistemes de signatures múltiples i en tecnologia blockchain.

2.2.3 Problemes criptogràfics en grups additius

Els problemes criptogràfics en grups additius són importants perquè proporcionen la base per a molts dels sistemes criptogràfics més utilitzats actualment. Ofereixen seguretat robusta, privacitat, eficiència en l'ús de recursos, i la capacitat de resistir atacs, fet que els fa crucials en la protecció de dades i en la seguretat de les comunicacions digitals.

Si aquests problemes són difícils de resoldre, l'impacte potencial de l'atac a un sistema criptogràfic basat en aquests problemes també serà molt alt, oferint una seguretat robusta contra intents de comprometre'l.

La definició dels grups additius feta a l'apartat anterior se'n desprèn un seguit de problemes de càlcul per a un adversari $\tilde{A}$ en un grup additiu $(\mathbb{G}_1 : +)$ enumerats per Joseph H. Silverman, en el seu llibre "Advanced Topics in the Arithmetic of Elliptic Curves" [9] i utilitzat com a base dels protocols criptogràfics moderns:

Problema 1: Problema del logaritme discret (Discret Logarithm Problem (DLP)): és costós per $\tilde{A}$ trobar $n \in \mathbb{Z}_q^*$ tal que $Q = n \cdot P$, donats dos elements del grup P i Q.

Problema 2: Problema de càlcul de Diffie-Helman (Computational Diffie–Hellman (CDHP)): és costós per $\tilde{A}$ calcular $a \cdot b \cdot P$, donats $P, a \cdot P, b \cdot P$ amb $a, b \in \mathbb{Z}_q^*$.

Problema 3: Problema decisional de Diffie-Helman (Divisible computation Diffie–Hellman problem (DCDHP)): és costós per $\tilde{A}$ decidir quan $c \equiv (a \cdot b) \bmod q$, donats $P, a \cdot P, b \cdot P, c \cdot P$ amb $a, b, c \in \mathbb{Z}_q^*$.

Problema 4: Problema de càlcul d'inversos de Diffie-Helman (Inverse computational Diffie–Hellman problem (Inv-CDHP)): donat $a \in \mathbb{Z}_q^*$ i tenint $P, a \cdot P$ és costós calcular $a^{-1} \cdot P$.

Problema 5: Problema de càlcul de l'arrel de Diffie-Helman (Square computational Diffie–Hellman problem (Sqr-CDHP)): donat $a \in \mathbb{Z}_q^*$ i tenint P,$a \cdot P$ és costós calcular $a^2 \cdot P$.

Definició: La Bilinealitat de Diffie-Helman (Bilinear computational Diffie–Hellman problem (BDHP)) a $(\mathbb{G}_1, \mathbb{G}_2, e)$ es defineix com: donats $(P, a \cdot P, b \cdot P, c \cdot P)$ per alguns $a, b, c \in \mathbb{Z}_q^*$, es calcula $v \in \mathbb{G}_2$ tal que $v = e(P, P)^{abc}$.

2.2.4 Esquema de signatura curta Zhang, Safavi, and Susilo per aparellaments bilineals

L'esquema de signatura Zhang, Safavi, and Susilo (ZSS) [10] es descriu com:

1. **ParamGen**: els paràmetres del sistema són $\{\mathbb{G}_1, \mathbb{G}_2, e, q, P, H()\}$
2. **KeyGen**: Seleccionar aleatòriament $x \in_R \mathbb{Z}_q^*$ i calcular $P_{pub} = x \cdot P$.

   P_{pub} és la clau pública i x és la clau privada.
3. **Signatura**: Donada una clau secreta x, i un missatge m, el càlcul $S = (H(m) + x)^{-1} \cdot P$ és la signatura.
4. **Verificació**: Donada una clau pública P_{pub}, un missatge m, i una signatura S, la verificació consisteix en comprovar que:

$$\begin{aligned} e(H(m) \cdot P + P_{pub}, S) &= e(H(m) \cdot P + x \cdot P, (H(m) + x)^{-1} \cdot P) \\ &= e((H(m) + x) \cdot P, (H(m) + x)^{-1} \cdot P) \\ &= e(P, P)^{(H(m)+x)\cdot(H(m)+x)^{-1}} \\ &= e(P, P) \end{aligned} \tag{2.1}$$

L'interès d'aquesta signatura és que permet validar signatures ofuscades, és a dir, es pot ofuscar la signatura multiplicant-la per un factor aleatori, de manera que no sigui reconeixible, però que passa la verificació si P s'ofusca amb el mateix valor. És a dir:

- Si es té una signatura S i es multiplica per un factor aleatori $b \in \mathbb{Z}_q^*$
- Es pot comprovar que:

$$\begin{aligned} e(H(m)\cdot P + P_{pub}, b\cdot S) &= e(H(m)\cdot P + x\cdot P, b\cdot (H(m)+x)^{-1}\cdot P) \\ &= e((H(m)+x)\cdot P, b\cdot (H(m)+x)^{-1}\cdot P) \\ &= e(P, b\cdot P)^{(H(m)+x)\cdot (H(m)+x)^{-1}} \\ &= e(P, b\cdot P) \end{aligned} \tag{2.2}$$

Qualsevol podrà comprovar la signatura, encara que estigui ofuscada, si es té P ofuscat amb el mateix factor. Això permet oferir a un tercer poder validar una signatura sense desvetllar el seu valor.

2.2.5 Esquema de compromís

Un protocol de compromís com el descrit per Pederson en l'article "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing" [16] es basa en el problema del logaritme discret i les propietats de l'estructura algebraica d'un grup cíclic. En aquest esquema, el compromís es fa en dues parts: el valor compromès i un valor aleatori que garanteix l'ocultació. La idea és que una persona pugui comprometre's a un valor sense que ningú el pugui saber fins que decideixi revelar-lo.

Consisteix en dues fases, una fase de compromís i una fase de revelació. Cal que els participants en un conjunt de paràmetres públics:

- $\mathbb{G}$ d'ordre q, on q és primer.
- P generador de $\mathbb{G}_q$.
- $p = 2\cdot q + 1$.
- l'únic subgrup d'ordre q de $\mathbb{Z}_p^*$.
- $x \in_R \mathbb{Z}_q$ que és un element x escollit aleatòriament de manera uniforme.

El procediment és:

- L'usuari tria $x \in_R \mathbb{Z}_q$ i calcula $h = (P^x \bmod p)$.
- L'usuari manté secret el valor de x i fa públics els valors de p, q, P, h.
- En la **fase de compromís** que es denota $C = commit(a, r)$ per fer el compromís de $a \in \mathbb{Z}_q$, l'usuari tria un $r \in_R \mathbb{Z}_q$ i calcula el compromís $C = (P^a \cdot h^r \bmod p)$.

- En la **fase de revelació,** per obrir el compromís C, l'usuari revela a i r, i el verificador comprova que $C = (P^a \cdot h^r \bmod p)$

L'esquema de compromís de Pederson és útil en moltes aplicacions criptogràfiques, com per exemple:

- **Protocols de votació**: On un votant es compromet amb el seu vot però no el revela fins a una data posterior.
- **Protocols de coneixement nul**: On es poden demostrar propietats d'un valor sense revelar el valor mateix.
- **Autenticació i proveïment de proves**: Per garantir que una persona coneix una informació secretament compromesa sense revelar-la directament.

2.3 Sistemes d'autentificació

Des de l'inici de l'era digital, els Sistemes Informàtics (SI) han tingut la necessitat d'identificar la persona o SI que es troba a l'altre costat d'una comunicació. Aquesta identificació es coneix com autenticació. En la literatura existent s'han trobat diferents tipus d'autenticació en el món digital que es poden classificar en els següents grups per ordre cronològic d'aparició.

2.3.1 Autenticació independent

Des de l'aparició dels sistemes operatius de temps compartit, als inicis de la dècada dels 60 [17], on diversos usuaris poden accedir i utilitzar els recursos del sistema de manera simultània, cada SI posseeix la seva base de dades d'usuaris on es guarden Dades Privades (DP) de cada usuari [18], el mètode d'*autenticació* i un *secret* (que pot ser una paraula clau, dades biomètriques o quelcom que sabem/tenim). Aquest escenari es pot veure a la Figura 2.5. Per l'explicació farem referència a Credencials (creds) quan ens referim a la combinació *nom/secret*.

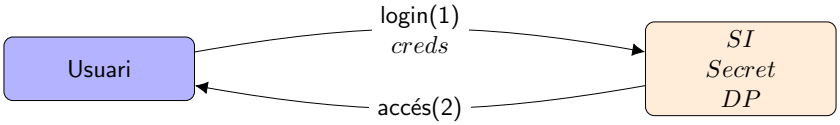

Figura 2.5: Autenticació independent.

Explicació de la Figura 2.5 :

(1) **Petició d'accés i autenticació**: L'*usuari* fa la petició d'accés (*login*), proporcionant les credencials *nom/secret*.

(2) **Validació i accés**: Després que l'SI comprova les *credencials* amb la seva base de dades, permet l'accés a l'*usuari* (*accés*).

Aquest tipus d'autenticació té un seguit d'avantatges i inconvenients:

Avantatges:

- És un sistema fàcil d'implementar, només cal guardar una taula amb la correspondència *nom/secret*.
- El *secret* està en forma de paraula clau.
- És intuïtiu, l'usuari té un *nom* i un *secret*, amb això en té prou per autenticar-se.

Inconvenients:

- Si el *secret* es fa públic qualsevol pot suplantar a l'*usuari*.
- En augmentar el nombre de serveis fa que l'*usuari* hagi de gestionar molts *noms/secret*, ja que no sempre es pot utilitzar el mateix *nom*, i a més, és aconsellable fer servir diferents *secrets* per cada *nom*.
- D'altra banda, diversos SI posseeixen dades privades que no sempre estan tan ben custodiades com caldria, i sovint hi ha fugues de dades personals[2][3][4][5][6].

2.3.2 Proveïdor d'autenticació o Identity Manager

Veient els inconvenients que presenta el mètode d'autenticació independent, durant la dècada dels 2000 apareixen terceres parts com Google (Google Id)[7], Facebook (Facebook login)[8] o OpenId[9] que ofereixen als diferents SI, la delegació de l'autenticació, aquests són els IDM.

Apareixen nous estàndards com el Security Assertion Markup Language (SAML)[10] o el Open Authorisation (OAuth)[11] que permeten a una entitat delegar i confiar en l'autenticació feta per un tercer (veure Figura 2.6).

Aquesta solució permet tenir un únic usuari amb una única paraula clau per a diversos sistemes. No obstant això, no resol el problema de la custòdia de la informació privada, que segueix sent fora del control de l'usuari. L'IDM genera un token, que és un conjunt de bytes amb el format del protocol ja siguin SAML o OAuth, té un temps de validesa, i amb la presentació d'aquest token a l'SI final, permet l'accés.

Les especificacions d'un IDM estan definides en el document "A framework for identity management – Part 1: Terminology and concepts" [21].

[2] URL: https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/2012-linkedin-breach-117-million-emails-and-passwords-stolen-not-6-5m.
[3] URL: https://www.bbc.com/news/technology-24740873.
[4] URL: https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/500-million-yahoo-users-affected-by-data-breach-password-change-recommended.
[5] URL: https://www.bbc.com/news/technology-37232635.
[6] URL: https://www.digitaltrends.com/news/facebook-data-leak-267-million-users-affected/.
[7] URL: https://developers.google.com/identity.
[8] URL: https://developers.facebook.com/docs/facebook-login.
[9] URL: https://openid.net/.
[10] URL: https://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html.
[11] URL: http://www.rfc-editor.org/rfc/rfc6749.txt.

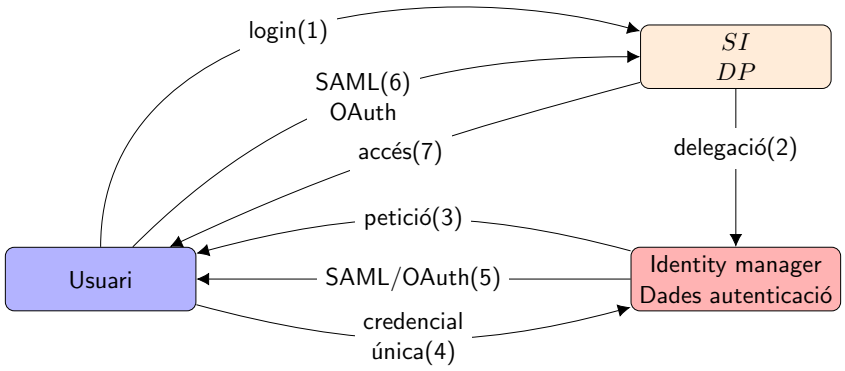

Figura 2.6: Autenticació amb IDM.

Explicació de la Figura 2.6 :

(1) **Petició d'accés**: L'usuari fa la petició d'accés (*login*).

(2) **Delagació**: L'SI delega l'autenticació redirigint l'usuari cap a l'IDM (*delegació*) si no ho ha fet recentment. Si ja ho ha fet passa al pas 6.

(3) **Petició de credencials**: L'IDM fa la petició de credencials a l'usuari (*petició*).

(4) **Autenticació**: L'usuari proporciona la *credencial ùnica* a l'IDM.

(5) **Autorització**: L'IDM comprova la *credencial ùnica* amb la seva base de dades i genera un token, ja sigui SAML o OAuth que retorna a l'usuari. (*SAML/OAuth*)

(6) **Prsentació de l'autorització**: L'usuari presenta el token generat per l'IDM a l'SI (SAML o OAuth).

(7) **Accés**: Després que l'SI comprova token i permet l'accés a l'usuari (*accés*).

Aquest tipus d'autenticació té un seguit d'avantatges i inconvenients:

Avantatges:

- És un sistema intuïtiu. L'usuari té un nom i un secret, amb això en té prou per autenticar-se.
- Les credencials són úniques per tots els serveis.
- Aquest sistema permet el Single Sign On (SSO), només ens hem d'autenticar un cop si diversos serveis fan servir el mateix IDM.
- I, finalment, les dades d'autenticació són úniques i estan en un sol lloc.

Inconvenients:

- És un sistema més complex d'implementar: apareix un tercer actor, augmenta el nombre d'intercanvis d'informació i cal estar d'acord sobre el tipus de token a utilitzar.
- A més té dependència d'aquest tercer, que si no està disponible es perd l'accés als serveis, temporalment o definitivament si el tercer desapareix.

- Aquest tercer passa a tenir un gran poder, provoca centralització i a més té informació sobre quins serveis utilitza un usuari, el que li permet confeccionar un perfil de cada usuari.
- També, si el secret es fa públic qualsevol el pot suplantar, a tots els serveis alhora.
- I finalment, les dades privades segueixen estan en les bases de dades dels diferents SI i no sempre estan custodiades com caldria. Sovint hi ha fugues de dades personals.
- Al ser mes complexes i necessitant una interacció en temps real de cada un del actors provoca molts cop fluxos vulnerables, com la usurpació d'identitats en OAuth[12] o SAML[13], o bypass de l'autenticació SAML[14].

2.3.3 Credencials basades en atributs verificables

També a principis dels anys 2000 apareixen les credencials basades en atributs verificables. L'enfocament de les credencials basades en atributs verificables busca preservar la privadesa de les dades, evitar la seva disseminació a la xarxa i traslladar la custòdia de les dades a l'usuari. Es basa en unes Entitats de Confiança (EC) que generen proves criptogràfiques irrefutables sobre atributs d'un usuari. Aquestes proves s'anomenen $\mathcal{AV}$. Solen ser signatures digitals, a la Taula 2.1 es pot veure la columna CPP que indica quins algoritmes de signatura utilitza cada solució. L'usuari pot, a posteriori, utilitzar els $\mathcal{AV}$ per provar la validesa dels atributs a un tercer. En la Figura 2.7 es pot veure l'arquitectura d'aquesta solució.

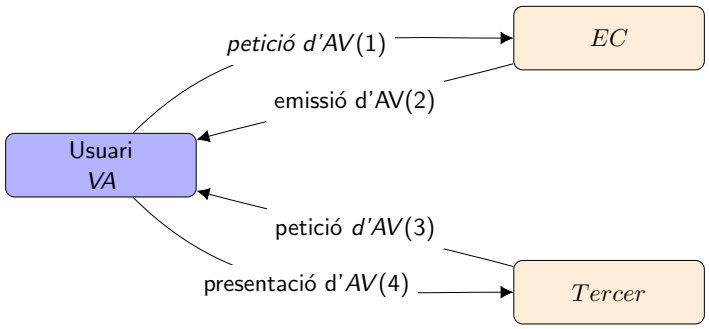

Figura 2.7: Autenticació amb $\mathcal{AV}$.

Explicació de la Figura 2.7 :

(1) **Autenticació i petició de generació de l'$\mathcal{AV}$**: L'usuari s'autentica amb un dels mètodes vistos en els apartats anteriors, ja sigui l'autenticació independent (Apartat 2.3.1) o l'autenticació amb l'IDM (Apartat 2.3.2), contra l'EC i fa la petició d'un atribut verificable.

[12] URL: `https://www.darkreading.com/application-security/oauth-flaw-exposed-millions-airline-users-account-takeovers`.
[13] URL: `https://www.darkreading.com/cloud-security/saml-flaw-lets-hackers-assume-users-identities`.
[14] URL: `https://projectdiscovery.io/blog/github-enterprise-saml-authentication-bypass`.

(2) **Comprovació i generació de l'$\mathcal{AV}$**: L'EC comprova que l'usuari té aquest atribut, i genera una prova irrefutable que és certa i la retorna a l'usuari. L'usuari emmagatzema la prova per una posterior utilització. Aquesta prova pot ser per exemple una signatura digital.

(3) **Petició de la presentació d'un $\mathcal{AV}$**: Un tercer fa la petició d'un atribut a l'usuari.

(4) **Presentació de l'$\mathcal{AV}$**: L'usuari mostra l'$\mathcal{AV}$ que correspongui a la petició. I pot comprovar que ha estat generada per una EC. Seguint l'exemple, validant la signatura digital.

Aquest mecanisme té un seguit d'avantatges i inconvenients:

Avantatges:

- L'usuari és qui custodia els seus $\mathcal{AV}$.
- Els proporciona a qui i quan vol. Només cal que aquest "qui" pugui verificar-lo.

Inconvenients:

- La falta d'un estàndard consolidat
- La poca adopció, encara hi ha pocs sistemes que l'implementin.

L'ús de la tecnologia Privacy-preserving Attribute Base Credencials (Privacy-ABC) que descriu Jan Camenish a [23] permet fer assercions (atestats) sobre identitats tot preservant la privacitat. Existeixen dos projectes fonamentats sobre Privacy-ABC: Idemix [24] i U-prove [73]. Una bona anàlisi comparativa sobre les capacitats d'aquesta tecnologia en les seves diferents implementacions es pot trobar a [14] d'on s'ha extret i ampliat la Taula 2.1.

2.4 El sistema d'autenticació autogestionada

En l'Apartat 2.3 s'han enumerat un seguit de solucions d'autenticació que presenten una sèrie d'inconvenients pels objectius de la recerca: d'una banda, la pèrdua del control per part de l'usuari i la disseminació a la xarxa tant de les dades d'autenticació com de les dades privades. Per una altra, la centralització de poder sense garantir ni preservar la privacitat, una certa facilitat de suplantació de la identitat, i finalment, presenta punts febles, o dit d'altra manera, punts valuosos on centrar els esforços d'atac. Així és com apareix, a principis del 2012 [74], el nou concepte de l'autenticació autogestionada o Self Sovereign Identity (SSI) en anglès, basada en atributs verificables, que pretén oferir una "identitat portàtil" per a qualsevol persona, organització o dispositiu que no depèn de cap autoritat centralitzada, que mai no es pugui suplantar, i a més on l'usuari té el control, en cada moment, del que vol desvelar i a qui. En un sistema "Self Sovereign", els usuaris existeixen independentment dels serveis. Christopher Allen resumeix els principis de l'SSI en 10 punts [75]:

- **Existència**: els usuaris han de tenir una existència independent. El cor de tota SSI es basa en última instància en el "Jo". L'objectiu d'una SSI ha de ser simplificar el procés de fer públic i accessible alguns aspectes d'aquest "Jo", que ja existeix.

Propietat → Esquema ↓	CPP[1]	UK[2]	UF[3]	UT[4]	CF[5]	BC[6]	UC[7]	SB[8]	KYC[9]
U-Prove [73]	$\mathcal{ECC}$	✗	No provat	No provat	✗	✗	✓	✗	✓
[25]	$\mathcal{ECC}$	✓	✓	No provat	✗	✗	✓	✓	✗
Idemix [24]	$\mathcal{RSA}$	✓	✓	✓	✓	✓	✗	✗	✓
Indy-anoncreds [89]	$\mathcal{RSA}$	✓	✓	✓	✓	✓	✗	✗	✓
[26]	$\mathcal{ECC}$	✓	✓	No provat	No provat	✓	No provat	✗	✗
[14]	$\mathcal{ECC}$	✓	✓	✓	✓	✓	✓	✓	✗

[1] **Protocol de presentació de credencials**: Fa referència a l'algoritme criptogràfic que es fa servir: $\mathcal{ECC}$ fa referència a l'algoritme de signatura amb corbes el·líptiques [27], i $\mathcal{RSA}$ a l'algoritme de signatura proposat per Rivest, Shannon i Adlman a [28].
[2] **Unlinkability**: propietat que fa referència a la incapacitat de poder enllaçar la identitat a un atribut.
[3] **Unforgeability**: propietat que fe referència a la incapacitat de poder crear una prova falsa.
[4] **Untraceability**: propietat que fa referència a la incapacitat de poder enllaçar l'ús d'unes mateixes credencials en diferents usos d'aquestes en diferents àmbits.
[5] **Confidencialitat**: propietat que fa referència a la capacitat d'emmascarar un atribut donat a l'hora de fer l'emissió o la presentació d'aquest.
[6] **Blockchain**: denota si existeix una implementació en Blockchain.
[7] **User Centric**: denota si l'usuari té el control de les credencials.
[8] **Self-blindable**: denota si l'usuari pot emmascarar informació.
[9] **Known Your Customer**: denota si el consumidor de les credencials pot enllaçar peticions en un mateix àmbit.

Taula 2.1: Comparativa d'esquemes de credencials Privacy-ABC

- **Control**: l'usuari ha de tenir el control de la seva identitat. Gràcies a algorismes criptogràfics segurs que garanteixen la vigència continuada de la seva identitat i el control de les seves dades personals.

- **Accés**: els usuaris han de tenir accés a les seves pròpies dades. Un usuari sempre ha de ser capaç de recuperar fàcilment totes les dades de la seva identitat.

- **Transparència**: els sistemes i algorismes han de ser transparents. Els sistemes que s'utilitzen per administrar i operar una xarxa SSI han de ser oberts. Els algoritmes han de ser lliures, de codi obert, coneguts i el més independents possibles de qualsevol arquitectura en concret.

- **Persistència**: les identitats han de ser de llarga durada. Preferiblement, les identitats han de durar per sempre, o com a mínim, durant el temps que l'usuari desitgi. : la informació i els serveis sobre la identitat han de ser transportables. Les identitats no les ha de tenir un SI de tercers, encara que sigui un SI de confiança que s'espera que funcioni en l'interès de l'usuari.

- **Interoperabilitat**: l'ús de les identitats hauria de ser el més ampli possible. Les identitats són de poc valor si només funcionen en nínxols limitats.

- **Consentiment**: els usuaris han d'acordar l'ús de la seva identitat. L'intercanvi de dades només s'ha de produir amb el consentiment de l'usuari.

- **Minimalista**: s'ha de minimitzar la divulgació d'atestats. Quan es publiquen dades, la revelació hauria d'incloure la quantitat mínima necessària de dades per dur a terme la tasca que es vol.
- **Protecció**: cal protegir els drets dels usuaris. Quan hi ha un conflicte entre les necessitats de la xarxa SSI i els drets dels usuaris individuals, la xarxa hauria de posar-se al costat de la preservació de les llibertats i drets de les persones per sobre de les necessitats de la xarxa.

En resum, els principis proposats per Christopher Allen sobre SSI es fonamenten en una visió de descentralització, privacitat, control de l'usuari i seguretat, amb l'objectiu de construir sistemes de gestió d'identitat més justos, segurs i transparents. Implementant aquests principis, es pot empoderar als individus per gestionar la seva pròpia identitat de manera més eficaç, tot garantint una millor protecció de la seva privacitat en el món digital.

2.4.1 Arquitectura de la SSI

Els sistemes SSI es basen en el model de credencials verificables proposat pel W3C a [29]. Aquest model proposa diversos conjunts de definicions: conceptes, rols i elements.

Conceptes: Primer, cal definir els conceptes sobre els quals es basa tot el model

- **Identificador**: és una cadena de caràcters, cada entitat té un o un conjunt d'identificadors.
- **Atestat** o **claim**: és una asserció que algú fa sobre una identitat, o sigui es una manera d'implementar els $\mathcal{AV}$s.

Rols: Es defineixen un conjunt d'actors, entitats que interaccionen entre elles de manera segura i fiable.

- **Usuari** o **holder**: és una entitat independent, que controla i posseeix els seus identificadors.
- **Emissor** o **issuer**: és l'entitat que certifica un atestat.
- **Consumidor** o **inspector**: és l'entitat que verifica o consumeix l'atestat.
- **Agent**: és una combinació de funcionalitats i emmagatzematge que es comunica via missatges amb d'altres agents. La seva funcionalitat permet interactuar i actuar en nom d'un usuari a través de regles de negoci.

Elements: Es defineix també un conjunt d'elements.

- **Magatzem de material d'autenticació** (**moneder** o **wallet**): On l'usuari guarda el material criptogràfic per l'autenticació.
- **Registre distribuït d'identificadors** (R_{id}): Guarda la correspondència entre l'identificador i la informació associada a aquest, necessària pel funcionament del sistema.

– **Repositori d'atestats ($R_{atestats}$)**: On els agents guarden els atestats.

En aquest paradigma, el material d'autenticació es manté privat i les dades per permetre aquesta s'emmagatzemen de manera distribuïda en una base de dades accessible per tothom que fa de registre d'identificadors. Per tant, cada usuari tindrà un o més d'aquests identificadors, que s'anomenen DID (per a més detalls veure l'Apartat 2.4.2). La idea és que la comunicació es duu a terme amb protocols d'extrem a extrem que es fonamenten en el registre de DID (Figura 2.8).

Tant les dades d'autenticació (DA en la Figura 2.8) com les dades privades (DP en la Figura 2.8) són propietat de l'usuari i només s'intercanvien entre extrems. La custòdia de les dades és responsabilitat de l'usuari i és ell qui decideix quina informació vol compartir.

Aquesta solució es basa en la utilització de criptografia asimètrica. Quan una part A vol autenticar a una altra B, A envia una petició amb un nonce (un número aleatori que només ell coneix) a B. Aquesta petició es coneix com repte. Quan B rep el repte, calcula la signatura amb la seva clau privada i la retorna a A (resposta al repte). Així, A pot verificar la signatura (resposta al repte) amb la clau pública de B. Aquest és el mecanisme d'autenticació que s'utilitza en aquest sistema, sense la necessitat de tenir tercera part de confiança.

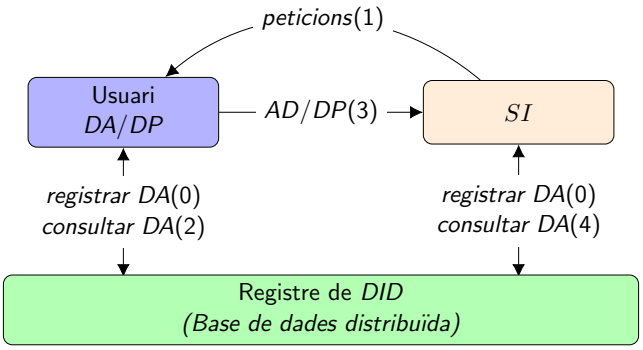

Figura 2.8: Autenticació amb SSI.

Explicació de la Figura 2.8 :

(0) **Generació i registre del DID**: L'usuari i l'SI generen el seu identificador descentralitzat (DID). Aquest identificador s'acompanya seva la clau pública, i ambdues dades es registren a la base de dades distribuïda. Aquest registre és una operació que només cal realitzar una vegada, en el moment de la creació del DID, abans que aquest sigui utilitzat per primera vegada. Un cop registrat, el DID serveix com a identificador persistent al llarg de la vida de l'usuari. Aquesta operació assegura la vinculació de l'usuari o l'SI amb el seu identificador de manera segura i permanent, evitant la necessitat de repetir-la en interaccions posteriors.

(1) **Consulta i petició de l'SI**: Un cop creat i registrat el DID, l'SI pot consultar el registre per obtenir informació sobre l'usuari. L'SI realitza una petició a l'usuari, que pot ser de diverses naturaleses, com ara una sol·licitud d'autenticació (DA) o una petició relacionada amb les dades privades (DP). Aquestes peticions es poden

considerar com a sol·licituds d'autorització per accedir a determinats recursos o per verificar l'autenticitat de l'usuari, depenent del context de la comunicació.

(2) **Validació de la petició per part de l'usuari**: Quan l'usuari rep una petició del SI, aquest primer haurà de verificar l'origen i la integritat de la sol·licitud. Això es realitza mitjançant criptografia asimètrica, utilitzant les claus públiques i privades per confirmar que la petició prové realment de l'entitat que se suposa que l'ha generat, i que no ha estat alterada en el procés de transmissió. Aquesta validació és crucial per garantir que l'usuari no sigui víctima d'atacs com la suplantació d'identitat o la manipulació de les dades.

(3) **Resposta de l'usuari a la petició**: Si la validació és correcta, l'usuari respon a la petició realitzant l'acció corresponent. Aquesta resposta pot consistir en l'enviament de (DA) o de (DP), depenent del tipus de petició rebuda. La resposta es formula d'acord amb els requeriments especificats a la petició inicial, garantint així que es mantingui la coherència entre la sol·licitud i l'acció de l'usuari. La resposta de l'usuari també pot incloure informació adicional, com ara dades d'autenticació o autorització, que permetin al SI procedir amb el seu procés de verificació.

(4) **Verificació de la resposta per part de l'SI**: Un cop l'usuari ha respost a la petició, l'SI torna a consultar el registre del DID per validar la resposta rebuda. Aquesta validació es fa mitjançant un nou procés de criptografia asimètrica, utilitzant les claus públiques per assegurar que la resposta no hagi estat alterada i que realment prové de l'usuari legítim. Un cop confirmada la resposta, l'SI pot procedir amb el seu processament i dur a terme l'acció corresponent, com ara concedir l'accés o actualitzar l'estat de la informació. Aquest mecanisme de doble validació (de la petició i de la resposta) és essencial per mantenir la seguretat i la confiança entre les parts implicades en el procés.

Aquest tipus d'autenticació té un seguit d'avantatges i inconvenients:

Avantatges:

- El sistema no depèn de cap tercer ja que el registre és distribuït.
- L'*usuari* és qui custodia les seves DP i les proporciona a qui i quan vol.
- A més, l'*usuari* és lliure d'utilitzar diferents DID i claus per diferents serveis.

Inconvenients: És complex d'implementar: cal disposar d'una base de dades distribuïda.

- La gestió dels DID i claus associades és complexa.
- La recuperació en cas de pèrdua de les claus privades que ens identifiquen és difícil.
- Hi ha poca adopció, encara hi ha pocs sistemes que l'implementin.

2.4.2 Distributed Identifiers

Com indica el seu nom, els DID són identificadors distribuïts que no depenen d'una entitat central per a la seva creació o gestió. Un grup de treball del W3C ha proposat un model per a la gestió dels DID [30].

Els DID no són únics, ja que cada usuari pot crear tants DID com desitgi, i cadascun d'aquests permet establir canals de comunicació xifrats i segurs amb altres persones, organitzacions o dispositius. Aquests identificadors no només serveixen per provar la identitat de l'usuari, sinó que també faciliten l'intercanvi segur d'atestats o credencials digitals.

El sistema no depèn d'una autoritat central per verificar els DID, sinó que cada usuari és responsable de registrar els seus DID en una base de dades pública i distribuïda. Això garanteix la seguretat, la transparència i la privacitat dels usuaris en el procés d'autenticació i intercanvi d'informació.

Segueix la sintaxi d'Universal Resources Notation (URN) especificada en l'RFC-8141[15]. Un DID és una cadena de text amb l'estructura següent:

did:example:123456789abcdefghi

Cada DID té un "document DID" que està publicat a la base de dades distribuïda. Un document DID és un document JavaScript Object Notation (JSON) [32, 33].

Aquest document publica un id (1), una llista de claus (2), una llista de mètodes d'autenticació (3) i una llista d'endpoints (4) per interactuar amb el propietari del DID. A banda, hi ha dos camps més que no són de negoci i serveixen per assegurar la integritat del sistema: (5) una marca de temps per a l'auditoria i (6) una signatura per assegurar la integritat. La Taula 2.9 mostra els elements estàndard d'un document DID [30].

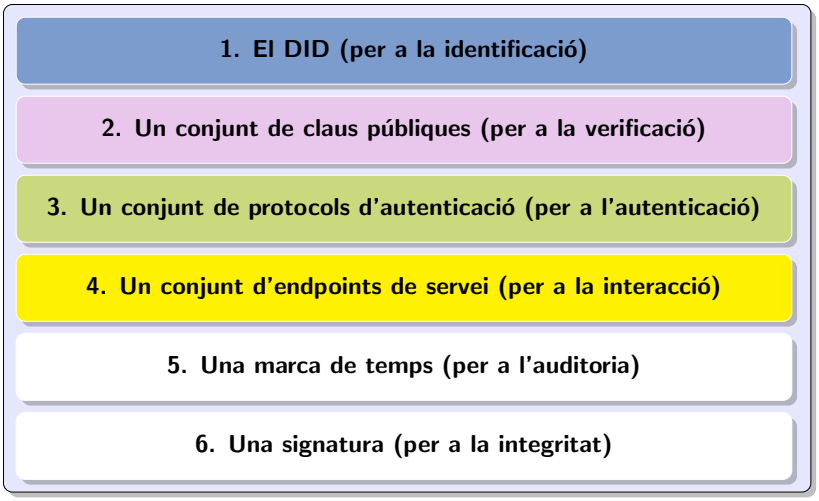

Figura 2.9: Elements estàndard d'un document DID.

Un exemple amb el mínim contingut es pot veure al Llistat 2.1.

```
{
"@context": "https://www.w3.org/ns/did/v1",
"id": "did:example:123456789abcdefghi",
"authentication": [{
        "id": "did:example:123456789abcdefghi#keys-1",
        "type": "RsaVerificationKey2018",
        "controller": "did:example:123456789abcdefghi",
```

[15] URL: http://www.rfc-editor.org/rfc/rfc8141.txt.

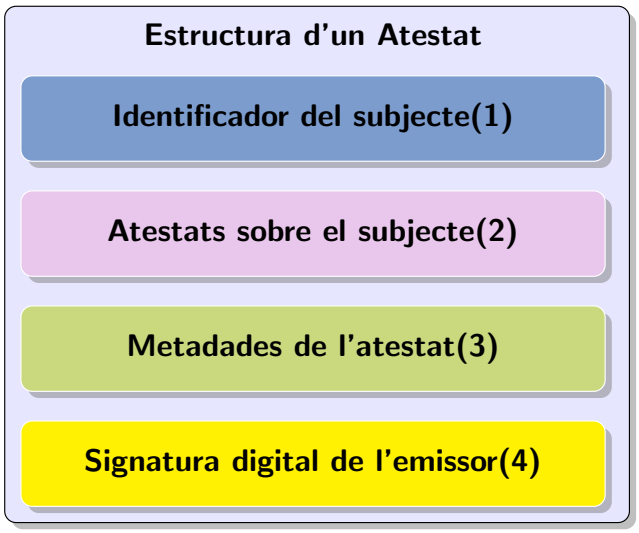

Figura 2.10: Estructura bàsica d'un VC. [76]

```
        "publicKeyPem": "-----BEGIN PUBLIC KEY...END PUBLIC KEY-----"
 }],
 "service": [{
        "id":"did:example:123456789abcdefghi#vcs",
        "type": "VerifiableCredentialService",
        "serviceEndpoint": "https://example.com/vc/"
 }]
}
```

Llistat 2.1: exemple mínim de DID

2.4.3 Atestats o Verifiable Claim

Els objectius de l'arquitectura d'atestat són: d'una banda, establir una arquitectura on l'usuari propietari d'un atestat té control total del seu identificador, d'on es guarda i de com s'utilitza. D'altra banda, busca reduir el frau creant una manera estàndard de compartir els atestats.

Finalment, vol assegurar al màxim la privacitat en el mecanisme de compartició dels atestats. Això inclou: separar la producció i el control d'un identificador de la producció d'atestats associada a aquest identificador, la separació del control de la compartició dels *atestats* de la seva creació, el desenvolupament d'estàndards per la interacció entre els diferents rols dins l'arquitectura i la reutilització de protocols ja existents quan sigui apropiat.

Es defineixen una sèrie de termes per definir amb propietat els atestats:

- **Atestat**: és una afirmació sobre un subjecte.
- **Subjecte**: és l'identificador que un holder proporciona i sobre la qual es fa l'atestat.
- **Atestat verificable** o **VC**: és un atestat del qual en podem verificar la seva autenticitat, la integritat i el no repudi.

L'estructura bàsica pels atestats verificables estableix els rols essencials dels diferents actors, les relacions entre ells i com interactuen. Aquests rols són:

- **Usuari**: és qui obté VC d'un emissor per presentar-los a un consumidor. L'usuari és sovint, però no sempre, el subjecte de l'atestat.
- **Emissor**: és qui genera VC per als usuaris.
- **Consumidor**: és qui demana VC als usuaris per autenticar-los.
- **Registre de DID**: és qui crea, verifica i manté el registre de DID.
- **Repositori**: és el magatzem de VC.

L'estructura bàsica d'un VC es mostra a la Figura 2.10.

L'identificador del subject (1) és el DID sobre el que es realitza l'atestat. Els atestats (2) són les afirmacions que l'emissor fa sobre el propietari de l'identificador, p.e, que és major d'edat. Les metadades (3) són informació sobre el document en si, com pot ser l'emissor o la data d'expiració d'aquest. I finalment, la signatura de l'emissor (4) que permet verificar l'autenticitat, la integritat i la validesa del document.

La Figura 2.11 mostra la visió general de l'arquitectura.

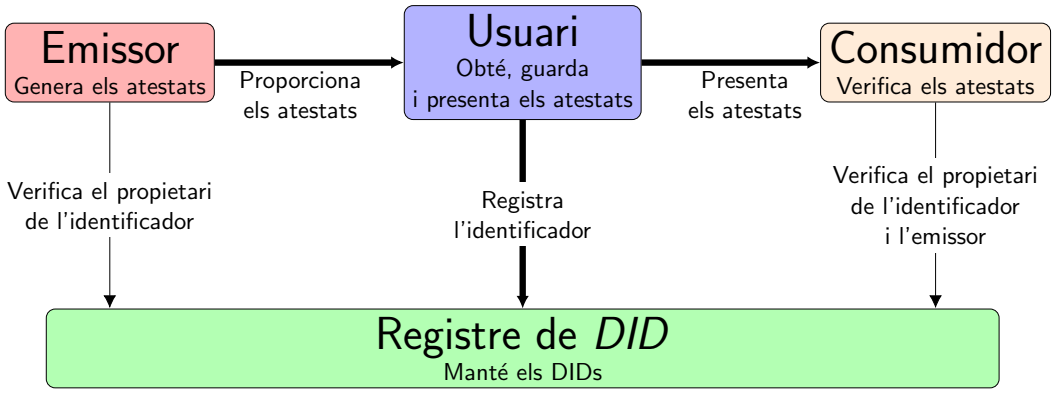

Figura 2.11: Arquitectura general dels VC, adaptació lliure de [29]

Tot seguit es descriu un cas d'ús. Na Mònica vol comprar un article en un lloc web que només està disponible per als majors d'edat, però ella no vol revelar ni el seu nom, ni quina edat té, ni la seva data de naixement. Els passos són: Pas 1, na Mònica (usuari) ha d'obtenir un identificador en l'SSI (Figura 2.12). Pas 2, na Mònica ha d'obtenir un atestat del registre civil (SSI_{rc}) que certifiqui que és major d'edat (Figura 2.13). Ara, na Mònica pot presentar aquest atestat a la botiga online (SSI_{botiga}) per poder dur a terme la compra (Figura 2.14). Com ho durà a terme pas a pas:

1. **Obtenció de l'identificador:** Aquest procés només involucra na Mònica i al *Registre de DID* (R_{did}). Gràficament es pot veure a la Figura 2.12 amb:

   (1) Na Mònica genera un DID, una parella de claus asimètriques i registra (*Registra*) el DID junt amb la clau pública al R_{did}.

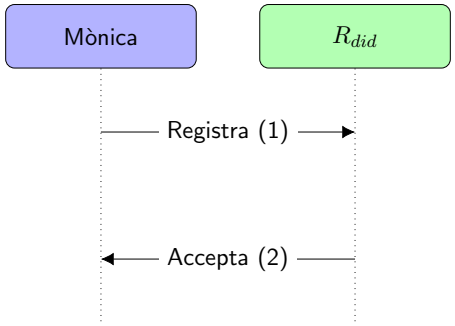

Figura 2.12: Na Mònica registra un DID amb la clau pública.

(2) El R_{did} respon (*Accepta*) conforme que el registre ha estat correcte.

2. **Obtenció d'un atestat que afirma que és major d'edat:** Aquest procés involucra na Mònica, el seu *agent*, l'SSI del registre civil (SSI_{rc}) i al *repositori d'atestats* ($R_{atestats}$) de l'*agent* de na Mònica. Gràficament es pot veure a la Figura 2.13 amb:

   (1) Na Mònica visita el web del registre civil (SSI_{rc}) per demanar un *atestat* (*Navegació web*) que digui que és major d'edat.

   (2) El seu *agent* fa una petició (*Petició d'un atestat*) l'SSI_{rc}.

   (3) L'SSI_{rc} verifica la *identitat* i la data de naixement de na Mònica (*Verificació*) amb la seva base de dades.

   (4) L'SSI_{rc} genera un *atestat* que diu que és major d'edat (*Generació*).

   (5) L'SSI_{rc} retorna l'*atestat* (*Emissió*) a l'*agent* de na Mònica.

   (6) L'*agent* mostra l'*atestat* rebut a na Mònica (*Mostra l'atestat*).

   (7) Ella pot veure l'*atestat* i si decideix que el vol guardar, li diu al seu *agent* (*Guarda l'atestat*).

   (8) L'*agent* persisteix l'*atestat* al $R_{atestats}$ (*Persisteix l'atestat*).

   (9) En resposta, l'*agent* rep una llista amb tots els atestats persistits (*Llista d'atestat*).

   (10) L'agent mostra aquesta llista a na Mònica (*Mostra llista d'atestats*). En cap cas l'*emissor* té coneixement de com es vol fer servir aquest *atestat* o a qui es presentarà.

3. **Na Mònica compra amb la presentació de l'atestat que afirma que és major d'edat:** Aquest procés involucra na Mònica, el seu *agent*, l'SSI del venedor ($SSI_{venedor}$) i al *repositori d'atestats* ($R_{atestats}$) de l'*agent* de na Mònica. Gràficament es pot veure a la Figura 2.14 amb:

   (1) Na Mònica visita la web d'un venedor ($SSI_{venedor}$) que requereix la majoria d'edat per accedir-hi (*Navegació*).

   (2) L'$SSI_{venedor}$ (actuant com a consumidor) fa una petició (*Petició*) d'una prova de majoria d'edat a l'*agent* de na Mònica.

   (3) L'*agent* busca una prova de majoria d'edat en l'$R_{atestats}$ (*Buscar*).

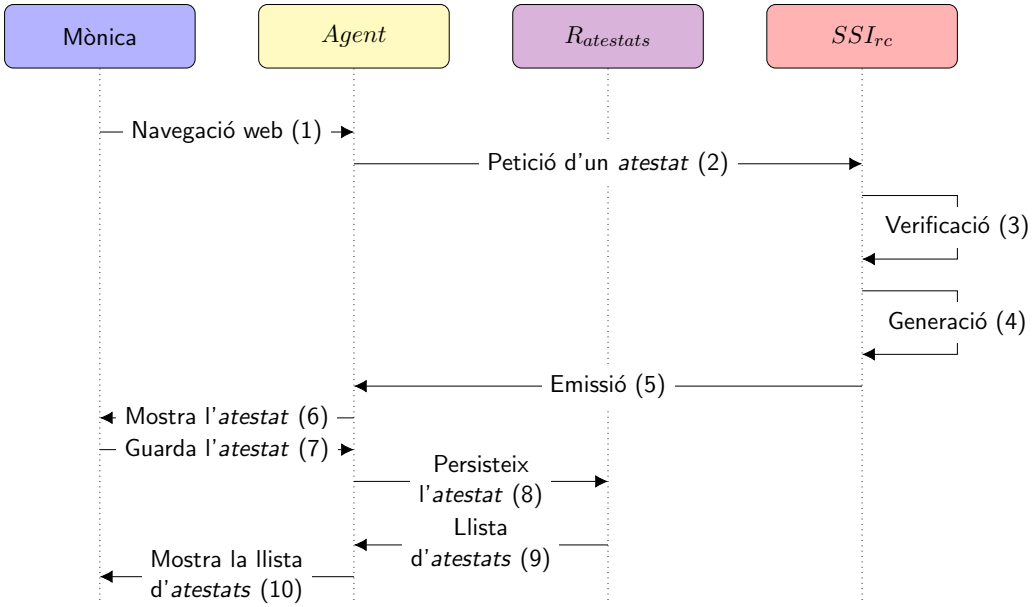

Figura 2.13: Na Mònica fa la petició d'un *atestat*.

(4) L'$R_{atestats}$ mostra tots els *atestat* que proven la majoria d'edat (*Mostra*).

(5) Na Mònica tria el que vol presentar (*Selecciona*).

(6) L'$R_{atestats}$ utilitza l'*atestat* com a prova de majoria d'edat (*Utilitza*).

(7) L'$SSI_{venedor}$ verifica l'atestat (*Verificació*).

(8) Un cop l'$SSI_{venedor}$ està segur que na Mònica té la majoria d'edat li permet l'accés al lloc web (*Dona*). En cap cas el *venedor* té coneixement de com es diu, quina edat o data de naixement té na Mònica, però pot estar segur que és major d'edat, en té una prova.

Aquest estudi s'ha centrat en aquesta última solució: l'autenticació basada en atributs, que garanteix major privadesa i control sobre les dades personals. A partir de l'inici dels anys 2000 s'han consolidat tot un seguit d'evolucions tecnològiques que han permès la implementació d'aquest tipus d'autenticació.

2.5 Tecnologia blockchain

La tecnologia distribuïda basada en blockchain està en posició de canviar dràsticament l'intercanvi d'informació digital, afirma Marvin Van Wingerdeen en la seva tesi "Blockchain-enabled self-sovereign identity" [34].

Internet afavoreix l'intercanvi d'informació de manera ràpida, i això fa que no estigui resolt el següent problema: cada cop que s'envia informació a algú, se li envia una còpia d'aquesta informació, no la informació en ella mateixa. Per tant, la transaccionalitat no està assegurada. Per això, cal que terceres parts actuïn com intermediaris per garantir aquesta transaccionalitat i aquestes terceres parts concentren, aleshores, un gran poder.

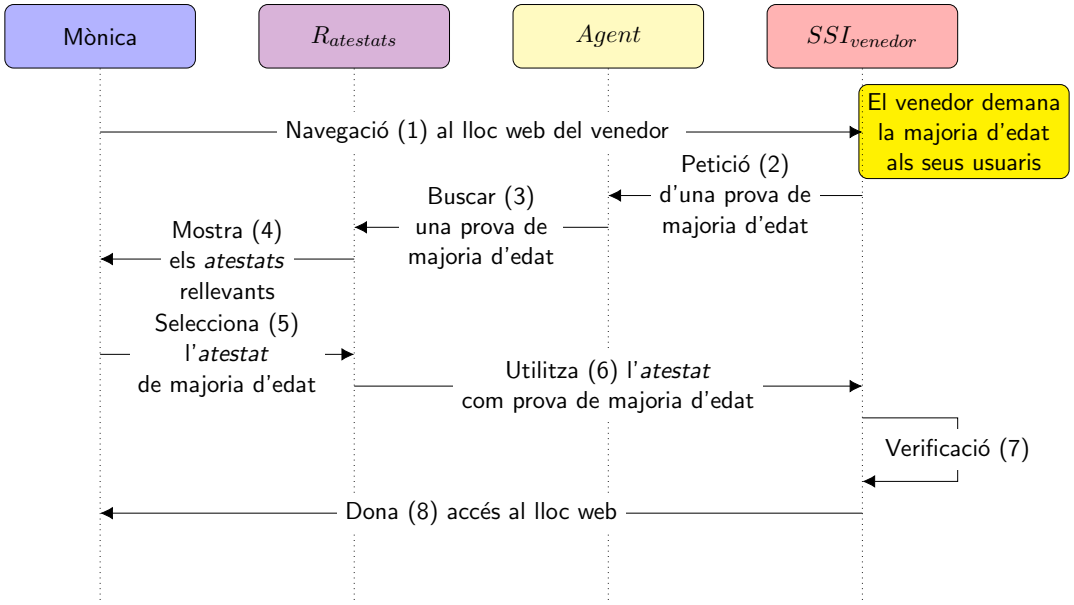

Figura 2.14: Na Mònica fa la presentació d'un atestat.

La tecnologia blockchain busca trencar aquesta concentració de poder, assegurant la transaccionalitat de manera distribuïda amb un protocol de consens entre tots els participants.

2.5.1 Definició de la tecnologia blockchain

En les tecnologies emergents, costa arribar a un consens per definir-les formalment i de forma acadèmica. Mougard a [36] descriu el potencial de la tecnologia blockchain des de tres perspectives:

- **Des d'un punt de vista de negoci**: la tecnologia blockchain és una xarxa on s'intercanvien valors digitalment peer-to-peer, és a dir, d'extrem a extrem, sense intermediaris.
- **Des d'un punt de vista legal:** la tecnologia blockchain és un protocol de xarxa que valida transaccions sense necessitat de confiar en un tercer.
- **Des d'un punt de vista tècnic:** la tecnologia blockchain és un suport (back-end) de base de dades públiques, distribuïdes i immutables.

La Distributed Ledger Technology (DLT) és el core de la tecnologia blockchain. Bàsicament, un Distributed Ledger (DL) és una base de dades distribuïda, mantinguda per un conjunt de nodes que executen un programari específic [37]. Dins la tecnologia blockchain els registres de les transaccions del DL s'agrupen en blocs que s'enllacen entre ells. Si tots els nodes que conformen la xarxa tenen els mateixos permisos i poden afegir blocs, es diu que és una permissionless DL. En canvi, si només un conjunt de nodes tenen la possibilitat d'afegir blocs, s'anomena permissioned DL. En l'àmbit de la tecnologia blockchain, a partir d'aquests dos tipus generals, prenen forma tres tipologies de DL cada una amb les següents característiques [38]: Els Public ledgers, són permissionless amb nodes anònims, i no cal cap grau de confiança entre nodes. També hi ha els Private ledgers que són permissioned, operats per una única entitat que ha de

tenir la total confiança dels membres de la xarxa. I finalment els Consortium/Hybrid ledger que són permissioned, però operats en aquest cas per un conjunt de nodes de confiança anomenats operadors. Els membres han de confiar en una majoria d'aquests operadors.

2.5.2 Tecnologia de capes de la tecnologia blockchain

La tecnologia blockchain té múltiples capes tecnològiques. Es poden definir tres capes bàsiques a la tecnologia blockchain basant-se en les descrites per Melanie Swan en l'article "Blockchain: Blueprint for a New Economy" [39]:

- **La capa de persistència**: formada per un DL públic amb registres que emmagatzemen les transaccions.
- **La capa de protocol**: composta pel programari de protocol que gestiona aquestes transaccions.
- **La capa d'aplicació**: que és la capa negoci, per exemple on s'implementa la criptomoneda en el cas de Bitcoin (BTC) [90]. Passem a descriure breument cada una de les capes en particular.

La capa de persistència: La criptografia és una peça fonamental de la tecnologia blockchain. La criptografia assegura la integritat de la cadena de blocs, permetent transaccions autenticades i assegurant un cert grau d'anonimat als membres de la xarxa.

Un dels algoritmes criptogràfics més utilitzats en els DL són els de la família Secure Hash Algorithm (SHA) en les seves versions 2 o 3), estandarditzat pel National Institute of Standards and Technology (NIST) [40]. Una funció hash és una funció sobre dades binàries amb una sortida de llargada fixa.

L'algoritme Secure Hash Algorithm versió 3 de 256bits (SHA3-256) agafa qualsevol tipus de dades binàries com a entrada – textos, imatges, vídeos o documents – i dóna com a sortida una cadena de text de 256 bits.

Aquest algoritme té una característica que el fa interessant: és relativament fàcil, a partir d'una entrada, obtenir la sortida, però és extremadament difícil, si no impossible, en un temps raonable, reconstruir l'entrada a partir de la sortida. A més, un petit canvi a l'entrada provoca que la sortida sigui totalment diferent. Per exemple, el SHA3-256 de "Hola món." és:

```
628440f275d0ecd3caaffe9fcbe911f96c90073f4b43490f47dedd9c780959b6
```

i només canviant el punt per un punt d'exclamació, el de "Hola món!" és:

```
14b268652b242819bb7a8f93b9d9760b5271c1f32f27b319ed00ee1507424cb5
```

La tecnologia blockchain registra transaccions en blocs, en una de les tipologies citades abans. Cada bloc està format per dos conjunts de dades: la capçalera del bloc i el contingut.

El contingut és una estructura d'arbre binari anomenat arbre de Merkle per Ralph Merkle al 1987 en un article anomenat "A Digital Signature Based on a Conventional Encryption Function" [41].

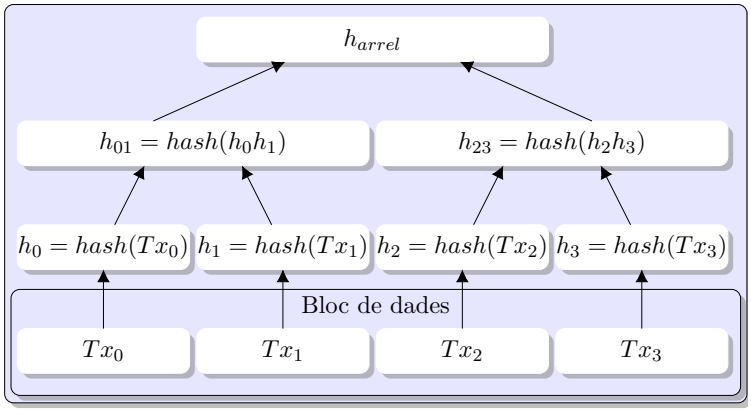

Figura 2.15: Estructura d'arbre de Merkle

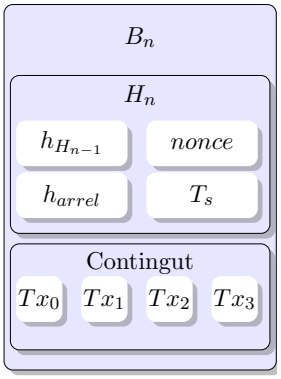

Figura 2.16: Estructura de dades d'un bloc Bitcoin.

Cada node de l'arbre conté el hash de la concatenació dels hashs dels seus fills, fins arribar a les fulles que són les transaccions i s'utilitza el hash de la transacció. Es pot veure de manera gràfica a la Figura 2.15.

El valor del hash de la capçalera del bloc es calcula a partir del hash de múltiples camps, dels quals els més importants són: el hash del bloc anterior dins la cadena ($H_{n_{-1}}$), el hash de l'arrel de l'arbre de Merkle del contingut del bloc actual (H_{arrel}), una marca de temps (T_s) i el nonce que és un número normalment calculat perquè compleixi certa condició matemàtica. Una representació visual del bloc es veu a la Figura 2.16. El contingut de cada bloc està format per transaccions (Tx_n).

Uns altres algoritmes criptogràfics àmpliament utilitzats en la tecnologia blockhain són els de la signatura electrònica. Aquests algoritmes de signatura es fonamenten en algoritmes de xifrat/desxifrat amb claus asimètriques i algoritmes hash. Un algoritme de xifrat és una transformació a partir d'una clau de xifrat sobre un text que el converteix en un garbuix de caràcters. A partir d'aquest garbuix només es pot reconstruir el missatge original si es coneix la clau de desxifrat; per tant, qui xifra i qui desxifra s'han de posar d'acord en quines claus utilitzen.

Existeixen dos tipus de xifrat: de clau simètrica i de clau asimètrica. El primer, fixa que la clau de xifrat i la de desxifrat sigui la mateixa. En canvi, en el segon cas, W. Diffie i M. Hellman l'any 1976 [67] van introduir el concepte de criptografia de clau pública o asimètrica, on distingeixen la clau de xifrat de la de desxifrat. En aquest article, es defineixen dos tipus de clau, la clau pública que és coneguda per tothom, i la clau privada que només és coneguda per la persona propietària de les claus. Aquestes claus estan construïdes per a que tot el que xifra l'una només es pugui desxifrar amb l'altra.

Aleshores, per signar digitalment unes dades – texts, imatges, documents o el que sigui – es calcula el seu hash i el resultat del xifrat d'aquest hash amb la clau privada serà la signatura. Per comprovar la signatura, es calcula el hash de les dades, es desxifra la signatura amb la clau pública del signant, i es comparen els dos valors. Si coincideixen es pot confiar en la veracitat de les dades i la seva autoria. Si no coincideixen, aleshores, o bé el signant no es correspon amb el de la clau pública utilitzada o bé s'han alterat les dades. Aquest mecanisme ofereix integritat, les dades no s'han alterat perquè els hashs coincideixen, i no repudi ja que la clau

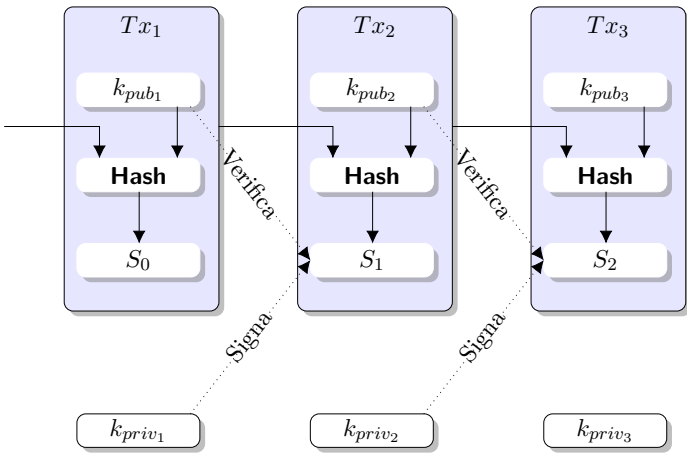

Figura 2.17: Encadenament de transacció de pagament simple en la Blockchain.

privada només la coneix el signant.

Un petit diagrama de l'encadenament de transaccions de pagaments simples es veu a la Figura 2.17. Cada transacció sol tenir la clau pública ($k_{pub_{rec}}$) del receptor de Tx_n, el hash de Tx_{n-1} (transacció prèvia), i la signatura de l'emissor(S_{em}), que es verifica amb la seva clau pública($k_{pub_{emi}}$) i està signada amb la seva clau privada($k_{priv_{emi}}$).

La capa de protocol: Un exemple de com s'emmagatzema una T_x de pagament simple amb el protocol Bitcoin (Figura 2.17) es mostra en el següent exemple: En Pau vol enviar una T_x a na Mònica.

Per iniciar T_x, en Pau necessita la clau pública de na Mònica i la seva pròpia clau privada. En Pau fa servir un programa específic anomenat moneder (wallet) per dur a terme la transacció.

El moneder crea una T_x on el receptor és la clau pública de na Mònica i la signa amb la clau privada d'en Pau.

El moneder emet aquesta transacció de forma global a la xarxa de membres. Cada membre de la xarxa comprova que en Pau pot fer aquesta T_x (p.e té suficient saldo).

La T_x, un cop acceptada, s'agrupa amb altres per formar un bloc. Un cop el bloc està llest es procedeix a "minar-lo" per provar la seva validesa i consensuar un nou estat de la Blockchain. En l'Apartat 2.5.3 es descriu el procés de validació dels blocs (conegut com a "minat") i el mecanisme de consens que s'utilitza.

La capa d'aplicació: La capa més alta és la capa d'aplicació, on s'implementen les criptomonedes (n'existeixen més de 11000 el febrer del 2025[16]). Les criptomonedes es creen generalment en base a dues aproximacions: comencen des de zero, sense que n'hi hagi en circulació o bé comencen amb un conjunt de monedes prefabricades.

Bitcoin va iniciar-se sense cap moneda en circulació. Les noves monedes es generen en validar blocs. A la data d'escriptura d'aquest document[17], aquell que afegeix un bloc a la cadena rep

[16] URL: `https://coinmarketcap.com/`.
[17] URL: `https://www.investopedia.com/terms/b/bitcoin-mining.asp`.

3.125 BTC com a recompensa pel seu esforç. Aquesta recompensa no és fixa i es va reduint en el temps. Aquesta reducció es produeix cada cop que s'assoleixen 210 000 blocs (cosa que passa aproximadament cada 4 anys), i la recompensa passa a ser la meitat del seu valor. Així ha passat dels 50 BTC al 2009, a 25 a partir del 2013, 12.5 durant el 2018 i a partir de l'11 de maig del 2020 val 6.25 BTC. A la darrera reducció a la meitat de Bitcoin l'abril de 2024, la recompensa va passar a 3.125 BTC.

El total i la freqüència de generació d'una criptomoneda venen definits pel protocol. En el cas del Bitcoin, hi ha un sostre de 21 milions de Bitcoins.

L'Ether [79], una altra criptomoneda, per la seva banda, començà com una moneda prefabricada pels seus desenvolupadors. Quan una moneda és prefabricada, una part del total de les monedes en circulació és assignada a una adreça abans de la creació de la Blockchain.

Diverses capes d'aplicació poden compartir la mateixa capa de protocol i la mateixa capa de persistència [39]. També diverses combinacions de capa d'aplicació i protocol poden compartir la capa de persistència.

2.5.3 Mecanisme de consens

Com ja s'ha comentat, existeix una xarxa de membres que suporta la persistència de les dades. Cada membre (o node) ha de tenir i estar d'acord amb l'estat de la blockchain. Cal arribar, doncs, a un consens sobre l'estat de la cadena de blocs. El mecanisme que defineix com cada node processa la transició d'estats s'anomena mecanisme de consens. D'acord amb [42], un mecanisme de consens ha de garantir tres premisses:

- **Seguretat**: que tots els nodes generin una sortida correcta, garantint un estat consistent de la cadena e blocs.
- **Participació**: que tots els nodes actius participants en el consens generin una sortida.
- **Tolerància a fallades**: que el protocol sigui capaç de recuperar-se davant la fallada d'un dels nodes participants al consens.

Existeixen múltiples mecanismes de consens, però generalment són de dos tipus: els basats en proves i els basats en validacions. Els basats en proves solen utilitzar-se en tecnologies blockchains permissionless.

En canvi, els basats en validacions s'utilitzen més en tecnologies blockchains privades. Seguidament es descriuen tres dels mecanismes més utilitzats en el consens basats en proves: Prof-of-Work (PoW), Prof-of-Stake (PoS) i Prof-of-Authority (PoA), aquesta darrera utilitzada sobretot en tenologies blockchains privades.

Prof-of-Work (PoW): Com el seu nom indica, els algoritmes PoW [90] es basen en l'ús de la potència de la Central Process Unit (CPU) i de la Graphics Process Unit (GPU) del node per provar el seu treball.

Per crear el bloc, tots els camps d'aquest són estàtics excepte el nonce. Per provar que un bloc és vàlid s'incrementa el nonce, i es calcula el hash a cada increment fins que aquest hash

calculat compleixi un determinat patró depenent del protocol. Per exemple, aquest patró pot ser que el hash, de llargada 32 bytes comenci amb quatre bytes a zero.

Com que cada bloc conté el hash del bloc anterior, si es canvia un bloc, cal recalcular el hash de tots els blocs que el segueixen, i degut a la dificultat de trobar el nonce que compleixi el patró determinat, aquesta tasca esdevé computacionalment i econòmicament impossible, en un temps raonable.

Com a recompensa pel seu treball, els miners reben una quantitat predeterminada de monedes per cada bloc validat. El principal inconvenient de les PoW és la quantitat d'energia que necessiten les CPU/GPU i que es consumeix en aquest procés.

Prof-of-Stake (PoS): En aquest cas [43], existeix un grup de validadors que proposen i voten cada bloc nou que s'afegeix. Cada vot es pondera amb la mida del dipòsit de cada participant.

Existeixen diversos tipus d'algoritmes de consens per a PoS, però hi ha dos tipus majoritaris[18]: Els chain-based i els Byzantine Fault Tolerance (BFT) style.

En la PoS basada en cadena (chain-based), cada cert temps (10 segons sol ser un període comú) l'algoritme tria pseudoaleatòriament un validador i li assigna el dret de crear un bloc. Aquest nou bloc ha d'apuntar a un bloc previ (normalment l'últim de la cadena més llarga) i d'aquesta manera tots els blocs convergeixen en una única cadena que va creixent.

En canvi, en la PoS del tipus BFT [44], cada validador, pseudoaleatòriament, té dret a proposar nous blocs, però l'acord sobre quin bloc és canònic es fa mitjançant un procés amb múltiples rondes on cada validador envia un "vot" per algun bloc específic a cada ronda i al final del procés tots (els validadors honestos i en línia) coincideixen definitivament sobre si un bloc donat forma o no part de la cadena.

Cal tenir en compte que els blocs estan igualment encadenats; la diferència clau és que el consens sobre un bloc només depèn del bloc en qüestió i no depèn de la longitud ni de la mida de la cadena posterior.

Prof-of-Authority (PoA): Aquest tipus de consens [45, 46] fixa que només un conjunt d'entitats autoritzades poden validar i afegir blocs a la cadena.

Les propietats més destacables d'aquest tipus de consens són l'eficiència energètica, ja que necessita menys recursos de càlcul, i una velocitat més alta processant transaccions utilitzant un mecanisme de consens basat en la identitat i la reputació dels validadors.

En una cadena PoA, les transaccions i els blocs són validats i aprovats per comptes coneguts com a validadors. En adjuntar una reputació a la identitat, els validadors estan incentivats a mantenir el processament de transaccions, ja que no volen que les seves identitats adquireixin una reputació negativa, que els penalitzi.

D'altra banda, PoA només permet l'aprovació de blocs no consecutius de qualsevol validador, el que significa que el risc de danys greus està centralitzat al node que autoritza. El principal inconvenient és una pèrdua de descentralització.

[18] URL: `https://ethereum.org/ca/developers/docs/consensus-mechanisms/pos/`.

2.5.4 Smart Contract

Els SC afegeixen una subcapa més en la capa d'aplicació. Afegeixen un grau més de complexitat a les transaccions bàsiques de la Blockchain que només permeten moviments de criptomonedes entre membres. Els SCs permeten la programació de transaccions.

Els SCs són essencialment trossos de codi amb els quals es pot assegurar la seva execució correcta, gràcies al mecanisme de consens de la tecnologia blockchain. Cada SC resideix a la cadena de blocs i té la seva adreça pública. A diferència d'un compte "normal", els SCs no posseeixen una clau privada, la seva integritat ve donada per formar part de la cadena de blocs.

La plataforma basada en SCs més popular és Ethereum[19]. Quan un SC es desplega sobre Ethereum, la seva adreça posseeix la capacitat de tenir certa quantitat d'Ether (la criptomoneda associada a Ethereum), espai d'emmagatzematge i el seu codi associat [47].

A Ethereum, per executar un contracte, cal gastar "gas". Aquest gas és com un combustible que equival a fraccions d'Ether. Això és així per evitar que un programa entri en un bucle infinit i col·lapsi la xarxa. Quan al contracte se li acaba el "gas", s'atura. Per tant, quan un usuari vol executar un SC, cal que enviï una transacció a l'adreça del contracte amb dues coses: la quantitat de gas màxima que està disposat a gastar i les dades d'entrada per l'execució del codi.

Un problema de seguretat amb és que no hi ha un mètode generalitzat de verificació formal [48] del codi dels SCs.

2.5.5 Gestió i persistència de les dades en la tecnologia blockchain

La DLT proporciona una manera altament segura de persistència de dades, però alhora, és costosa i lenta. Per això és important decidir què es guarda i què no es guarda a la cadena de blocs [50, 79]. Sovint s'utilitza com a registre immutable sobre recursos externs a la cadena de blocs, on es guarda la ubicació i el hash del recurs que es vol protegir.

Per definició, la tecnologia blockchain no té accés a recursos o fonts de dades externes. Tanmateix, els SCs necessiten dades de fonts d'aquestes. Els Oracles – uns agents que busquen i verifiquen dades en el món real per injectar-les a la cadena de blocs – són solucions fora de la cadena (off-chain) que recol·lecten dades pels SCs com il·lustren Azouvi et al. a l'article [49]. L'avantatge dels Oracles és que, en ser externs, poden tractar les dades de forma més eficient, però perdent el mecanisme d'integritat del consens.

A banda dels Oracles, una altra manera de gestionar dades en la tecnologia blockchain és el que es coneix com a emmagatzematge distribuït. Hi ha dos projectes en aquest sentit: Swarm[20] forma part de la pila de protocols d'Ethereum i l'InterPlanetary File System (IPFS)[21] que és una implementació independent. Ambdues estan inspirades en BitTorrent[22], i són solucions d'emmagatzematge distribuït que fan servir el paradigma Content Addressable Storage (CAS). D'aquesta manera, els SCs poden accedir a les dades a través del seu hash sense saber on ni com es guarden.

[19] URL: `https://ethereum.org/ca`.
[20] URL: `https://www.ethswarm.org/`.
[21] URL: `https://ipfs.tech/`.
[22] URL: `https://www.bittorrent.org/beps/bep_0003.html`.

2.6 La tecnlogia blockchain Ethereum

Ethereum és la tecnologia blockchain escollida i per aquesta raó s'aprofundirà en les característiques que ens interessen d'ella.

En el seu white paper [85], Vitalik Buterin introdueix Ethereum amb l'objectiu de superar les limitacions del llenguatge de programació de Bitcoin. La principal aportació d'Ethereum és la seva capacitat per oferir una màquina de Turing completa, la qual permet executar tota mena de càlculs, incloent-hi estructures de control com els bucles. A més, Ethereum introdueix el concepte d'estat dins de les transaccions, juntament amb una sèrie de millores respecte a l'estructura de la tecnologia blockchain original.

Ethereum es pot descriure com una tecnologia blockchain que integra un ordinador descentralitzat, amb el seu propi llenguatge de programació, i que permet l'execució d'aplicacions de manera descentralitzada i sense necessitat de permisos. Aquesta plataforma suporta l'execució de SCs i inclou un conjunt de regles criptogràfiques que s'executaran sota condicions específiques.

Encara que la tecnologia blockchain d'Ethereum té similituds amb la de Bitcoin, hi ha una diferència fonamental: mentre que els blocs de Bitcoin només contenen dades com el número de bloc, la dificultat, el nonce, la llista de transaccions, etc., els blocs d'Ethereum, a més d'aquesta informació, inclouen l'estat actualitzat de la cadena. Així, amb cada transacció, s'actualitza l'estat, aplicant els canvis sobre l'estat anterior.

Cada node dins de la xarxa d'Ethereum executa l'Ethereum Virtual Machine (EVM), que és responsable d'executar els SCs. Aquests contractes es tradueixen a codi compatible amb l'EVM i s'executen de manera descentralitzada en cada node de la xarxa [50]. Un dels llenguatges de programació més utilitzats per escriure smart contracts a Ethereum és Solidity[23].

2.6.1 Els comptes

L'estat d'Ethereum consisteix en un conjunt de comptes, on cada compte té una adreça de 20 bytes, i un conjunt de transicions d'estat. L'estat global és la relació entre adreces i estats dels comptes [50]. Ethereum suporta dos tipus de comptes: els comptes controlats externament (amb la corresponent clau privada) i els comptes controlats per un contracte [85]. Un compte Ethereum està format per quatre camps: (1) el nonce, (2) el balanç d'Ether, (3) el hash del codi del contracte i (4) l'arrel de magatzem.

1. **El nonce**: representa el nombre de transaccions enviades des d'una adreça particular o bé el nombre de creacions de contracte fetes per un compte i s'utilitza per garantir que cada transacció només pot ser executada una vegada.

2. **El balanç d'Ether**: és el nombre de Wei que posseeix l'adreça (Wei representa la fracció més petita d'Ether, un Ether (ETH) és igual a 10^{18} Wei). L'Ether s'utilitza com a moneda per pagar les comissions.

[23] URL: `https://soliditylang.org/`.

3. **El hash del codi del contracte**: és un hash Keccack-256 [51] (una variant del SHA3-256 que es diferencia de l'estàndard aprovat pel NIST [52] en el padding) del codi EVM associat a un compte, i que serà executat si rep una crida a la seva adreça.

4. **L'arrel del magatzem**: és un hash de 256 bits de l'arrel del Merkle tree que representa el contingut del compte. Els Merkle tree s'utilitzen per emmagatzemar totes les parelles (clau, valor) involucrades en l'ecosistema d'Ethereum.

2.6.2 Transaccions i missatges

Una transacció és una instrucció simple amb una signatura criptogràfica. Existeixen dos tipus de transaccions: les que tenen com a resultat crides i les que creen comptes. Una transacció es defineix com un paquet de dades signat que envia un compte controlat externament. Cada transacció està formada pel destinatari del missatge, una signatura que identifiqui el remitent, la quantitat d'Ether per enviar, un camp de dades opcional, i els valors `startgas` i `gasprice` [85, 50].

Els camps `startgas` i `gasprice` són vitals per combatre possibles actuacions malicioses i evitar el col·lapse de la xarxa. El gas és la unitat fonamental de càlcul. Cada transacció requereix una certa quantitat de càlcul, i `startgas` és la quantitat màxima de càlcul que una transacció pot consumir. Normalment s'utilitza 1 gas per un pas de càlcul més un fix addicional de 5 gas per cada byte d'emmagatzematge, encara que pot ser major i ve fixat per `gasprice` [85]. Com els miners són més recompensats si executen transaccions amb un `gasprice` més alt, cal que qui envia una transacció tingui en compte aquest fet si vol que la seva transacció sigui processada [50].

La funció de transició d'estat, que canvia els estats de l'emissor i del receptor en executar la transacció, comença verificant que la transacció sigui correcta (la signatura ha de ser vàlida i el nonce ha de correspondre al del compte de l'emissor). Si això és correcte, calcula les comissions com `startgas*gasprice`, resta aquesta quantitat del balanç de l'emissor i incrementa el nonce. La comissió es paga per bytes dins la transacció i la quantitat d'Ether sol·licitada es transfereix al receptor. Si el compte receptor no existeix, es crea, i si és un contracte, s'executa el seu codi. Si l'emissor no té prou quantitat d'Ether per la transacció o l'execució del codi, consumeix tot el gas; la funció de transició d'estat reverteix tots els canvis excepte el pagament de comissions als miners [85].

Un contracte pot enviar missatges a un altre contracte, és com una transacció però generada pel codi d'un contracte. Com una transacció, aquest missatge indueix a l'execució del codi del contracte.

2.7 Protocol d'autenticació amb credencials anònimes basat en atributs

Kalpanan Singh et al. proposen a [14] un protocol d'autenticació amb credencials anònimes basat en atributs, que preserva la privadesa i l'anonimat. S'ha escollit com a base pel desenvo-

lupament de la solució de la recerca realitzada. Aquesta secció fa una anàlisi en detall d'aquest protocol, buscant possibles millores que justifiquin aquesta recerca.

2.7.1 Arquitectura

En aquest sistema existeixen quatre tipus d'actors:

- **Usuari ($\mathcal{U}$)**: és qui vol utilitzar els serveis oferts pels proveïdors de serveis té un seguit de dades personals de les quals vol minimitzar l'intercanvi i preservar el control. També demana privacitat en les activitats que realitza i que no es puguin enllaçar.
- **Identity Validator ($\mathcal{IV}$)**: és el responsable de validar els atributs de la identitat que ell coneix i formen part del seu negoci. Un cas que s'utilitza en els exemples és el registre civil que coneix les dades privades d'un individu ja que és la seva funció. És conegut per tothom.
- **Certificate Provider ($\mathcal{CP}$)**: és qui genera credencials després de verificar la signatura d'un $\mathcal{IV}$ sobre els atributs anonimitzats i les seves proves sobre aquests.
- **Service Provider ($\mathcal{SP}$)**: és qui proveeix serveis a cada usuari interessat en els seus serveis i certificat pel $\mathcal{CP}$. L'$\mathcal{SP}$ verifica les credencials sense veure els atributs ni conèixer la identitat de l'usuari.

Poden existir diversos actors de cada tipus i la comunicació entre ells pot ser de dues maneres, offchain on les comunicacions tenen lloc sense guardar informació a la Blockchain, o onchain on les comunicacions requereixen interacció amb la Blockchain, per llegir o escriure dades.

Apareixen tres fluxos d'informació entre els diferents actors:

- $\mathcal{U}$ calcula el compromís dels atributs que vol certificar i els envia a $\mathcal{IV}$. Aquest darrer els verifica, ja sigui amb un document físic d'$\mathcal{U}$, amb la informació que ell posseïx o de la manera que sigui, assegurant-se que els atributs són certs. Seguidament, $\mathcal{IV}$ genera una signatura Elliptic Curve Digital Signature Algorithm (ECDSA) sobre el compromís dels valors d'atributs validats d'$\mathcal{U}$, i li envia. Amb la signatura d'$\mathcal{IV}$ sobre el compromís, $\mathcal{U}$ pot autenticar-se a la Blockchain.
- Per a què $\mathcal{CP}$ generi les *credencials* per $\mathcal{U}$, $\mathcal{U}$ envia el compromís dels valors dels atributs, la signatura de $\mathcal{IV}$ sobre aquest compromís i una prova de coneixement nul del tipus Non Interactive Schnorr Zero Knowledge Proof (NI-Schnorr) per cada valor. $\mathcal{CP}$ verifica la signatura d'$\mathcal{IV}$, les proves de coneixement nul NI-Schnorr per cada atribut utilitzat pel càlcul del compromís, verifica també l'atestat, signa el compromís amb ZSS modificat i finalment emet les credencials i les envia a $\mathcal{U}$.
- $\mathcal{U}$ presenta les credencials sense que $\mathcal{SP}$ tingui una identificació ni una verificació pública. $\mathcal{U}$ cega les credencial (generades per $\mathcal{CP}$), i s'autentica davant del $\mathcal{SP}$ onchain. El $\mathcal{SP}$ verifica si $\mathcal{U}$ té credencials de $\mathcal{CP}$ o no. Si és que sí se li permet l'accés, si és que no, avorta la comunicació.

– $\mathcal{U}$ utilitza les signatures ZSS modificades i l'esquema de self-credencials de Verheul's per ofuscar les seves credencials i les seves claus a partir de les credencials originals que $\mathcal{CP}$ li ha donat. Aquestes credencials cegues segueixen essent verificables i mantenen els drets de signatura de $\mathcal{CP}$.

– $\mathcal{U}$ fa servir les credencials cegues en les comunicacions amb la Blockchain. I així, ningú pot resseguir la seva activitat en la Blockchain, i els seus atributs resten protegits.

En cas de necessari, $\mathcal{SP}$ pot preguntar a $\mathcal{CP}$ que reveli els atributs d'identitat d'$\mathcal{U}$. $\mathcal{CP}$ envia una petició a $\mathcal{IV}$ per a què reveli la identitat real d'$\mathcal{U}$. D'aquesta forma es gestiona el control del atributs d'identitat d'$\mathcal{U}$ tant per part d'ell mateix com d'$\mathcal{IV}$.

2.7.2 Protocol

Generació de les claus

Cada actor genera un número aleatori secret $sk \in_R \mathbb{Z}_q$ i realitza una multiplicació escalar sobre la corba el·líptica per obtenir la clau pública $pk = sk \cdot P$. Tindrem doncs 4 parelles de claus $(sk_{\mathcal{U}}, pk_{\mathcal{U}})$, $(sk_{\mathcal{IV}}, pk_{\mathcal{IV}})$, $(sk_{\mathcal{CP}}, pk_{\mathcal{CP}})$, $(sk_{\mathcal{SP}}, pk_{\mathcal{SP}})$.

Emissió de la signatura del compromís ($\mathcal{U} \Longleftrightarrow \mathcal{IV}$)

Es fa offchain. $\mathcal{IV}$ verifica tots els atributs $a_1, a_2, \ldots, a_m$ i els seus valors $v_1, v_2, \ldots, v_m$ de $\mathcal{U}$, i registra $\mathcal{U}$ com a membre autoritzat. Tenim un grup cíclic $\mathbb{G}$ d'ordre q primer amb generadors $P_0, P_1, \ldots, P_n \in R(\mathbb{F}_{\shortparallel})$, el secret $sk_{\mathcal{U}}$ i la clau pública $pk_{\mathcal{U}} = sk_{\mathcal{U}} \cdot P_0$. L'intercanvi de missatges queda:

1. $\mathcal{U}$ tria un valor aleatori $r \in_R \mathbb{Z}_q$. El compromís C serà: $C = r \cdot pk_{\mathcal{U}} + (v_1 P_1 + v_2 P_2 + \ldots + v_n P_n)$.
2. $\mathcal{U}$ envia C i r a $\mathcal{IV}$.
3. $\mathcal{IV}$ comprova de manera fiable els atributs (p.e. presentació de documents).
4. $\mathcal{IV}$ signa el compromís amb l'algoritme $ECDSA$: $\sigma_{\mathcal{IV}} \leftarrow ECDSA - Sign(C, sk_{\mathcal{IV}})$.
5. $\mathcal{IV}$ envia $\sigma_{\mathcal{IV}}$, $pk_{\mathcal{IV}}$ a $\mathcal{U}$.

Emissió de credencials ($\mathcal{U} \Longleftrightarrow \mathcal{CP}$)

Es fa offchain. També es pot realitzar onchain. $\mathcal{U}$ s'autentica anònimament contra $\mathcal{CP}$ ja que $\mathcal{U}$ té la signatura del compromís C per part d'$\mathcal{IV}$. $\mathcal{U}$ envia la prova de coneixement nul NI-Schnorr dels valors del compromís C. $\mathcal{CP}$ verifica la signatura i la prova de correcció del compromís. Els passos són:

1. $\mathcal{U}$ té $C = r \cdot pk_{\mathcal{U}} + (v_1 \cdot P_1 + v_2 \cdot P_2 + \ldots + v_n \cdot P_n)$.
2. $\mathcal{U}$ calcula la prova de coneixement NI-Schnorr de C sobre EC de la següent forma:

- Tria $w \in_R \mathbb{Z}_q^*$
- Calcula $A = w \cdot pk_u$, $s = H(A)$ i $t = s \cdot r + w$
- Tria $w_i \in_R \mathbb{Z}_q^*$ per $i \in [1, n]$.
- Calcula $A_i = w_i \cdot P_i$, $s_i = \mathcal{H}(\mathcal{A}_\rangle)$ i $t_i = s_i \cdot v_i + w_i$.

3. $\mathcal{U}$ envia C, $\sigma_{\mathcal{IV}}$, (A,t,$r \cdot pk_{\mathcal{U}}$), (A_i,t_i,$v_i \cdot P_i$) a $\mathcal{CP}$

4. $\mathcal{CP}$ verifica les dades rebudes de la següent forma:

   - Verifica *ECDSA-verify*$(C, \sigma_{\mathcal{IV}}, pk_{\mathcal{IV}})$ és a dir: $e(H(C)P + pk_{\mathcal{IV}}, \sigma_{\mathcal{IV}}) \stackrel{?}{=} e(P, P)$
   - Si és vàlid verifica
     - $t \cdot pk_{\mathcal{U}} \stackrel{?}{=} A + r \cdot pk_{\mathcal{U}} \cdot s$
     - $t_i + P_i \stackrel{?}{=} A_i + v_i \cdot P_i \cdot s_i \ \forall i \in [1, n]$
     - $C \stackrel{?}{=} r \cdot pk_{\mathcal{U}} + \sum_{i=1}^{n} v_i \cdot P_i$
   - Si les verificacions són correctes calcula $M_{ZSS-Sign(C,sk_{\mathcal{CP}})} = \sigma_{\mathcal{CP}} = (H(C)+pk_{\mathcal{CP}})^{-1} \cdot pk_{\mathcal{U}}$

5. $\mathcal{CP}$ envia C, $\sigma_{\mathcal{CP}}$, $pk_{\mathcal{CP}}$.

6. $\mathcal{U}$ verifica que:

$$\begin{aligned} e(H(C)) \cdot P + pk_{\mathcal{CP},\sigma_{\mathcal{CP}}}) &= e((H(C) \cdot P + sk_{\mathcal{CP}} \cdot P), (H(C) + sk_{\mathcal{CP}})^{-1} \cdot pk_{\mathcal{U}}) \\ &= e((H(C) + sk_{\mathcal{CP}}) \cdot P, (H(C) + sk_{\mathcal{CP}})^{-1} \cdot pk_{\mathcal{U}}) \\ &= e(P, pk_{\mathcal{U}})^{(H(C)+sk_{\mathcal{CP}}) \cdot (H(C)+sk_{\mathcal{CP}})^{-1}} \\ &= e(P, pk_{\mathcal{U}}) \end{aligned}$$

Presentació de credencials ($\mathcal{U} \Longrightarrow \mathcal{SP}$)

El protocol proposat afegeix una característica addicional de privadesa per protegir les activitats d'$\mathcal{U}$ a la Blockchain mitjançant un esquema d'autocegat. Per això utilitza la signatura curta modificada ZSS i les autocredencials de Verheul's per ofuscar les claus utilitzades respecte a les entregades per $\mathcal{CP}$. Aquestes credencials ofuscades encara són verificables i mantenen la signatura de $\mathcal{CP}$. Els passos són:

1. $\mathcal{U}$ té $\sigma_{\mathcal{CP}}$, C, $sk_{\mathcal{U}}$, $pk_{\mathcal{U}}$.

2. $\mathcal{U}$ tria $b \in_R \mathbb{Z}_q^*$ com a factor d'ofuscació.

3. $\mathcal{U}$ calcula per una banda:

   - $sk'_{\mathcal{U}} = b \cdot sk_{\mathcal{U}}$, $pk'_{\mathcal{U}} = b \cdot sk_{\mathcal{U}} \cdot P$
   - $\sigma'_{\mathcal{CP}} = b \cdot \sigma_{\mathcal{CP}}$
   - $P' = b \cdot P$
   - $pk'_{\mathcal{CP}} = b \cdot pk_{\mathcal{CP}}$

- $C' = b \cdot H(C)$

4. $\mathcal{U}$ també calcula la prova de coneixement nul NI-Schnorr:
   - tria $r' \in_R \mathbb{Z}_q^*$
   - calcula:
     - $R' = r' \cdot P$
     - $s' = H(R')$
     - $t' = s' \cdot sk'_{\mathcal{U}} + r'$
5. $\mathcal{U}$ envia (R', s', t') per la prova NI-Schnorr i $\sigma'_{\mathcal{CP}}$, $pk'_{\mathcal{U}}$, $pk'_{\mathcal{CP}}$, P', C' per la validació de les credencials.
6. $\mathcal{SP}$ tria $\lambda \in_R \mathbb{Z}_q^*$ i calcula:
   - $\sigma = \lambda \cdot pk_{\mathcal{CP}}$
   - $pk''_{\mathcal{CP}} = \lambda \cdot pk'_{\mathcal{CP}}$
7. $\mathcal{SP}$ envia $pk''_{\mathcal{CP}}$
8. $\mathcal{U}$ calcula $pk'''_{\mathcal{CP}} = b^{-1} \cdot pk''_{\mathcal{CP}}$
9. $\mathcal{U}$ envia $pk'''_{\mathcal{CP}}$
10. $\mathcal{CP}$ si $\sigma \equiv pk'''_{\mathcal{CP}} \Rightarrow pk'_{\mathcal{CP}}$ és correcte:

$$\begin{aligned} pk'''_{\mathcal{CP}} &= b^{-1} \cdot pk''_{\mathcal{CP}} \\ &= b^{-1} \cdot \lambda \cdot pk'_{\mathcal{CP}} \\ &= b^{-1} \cdot \lambda \cdot (b \cdot pk_{\mathcal{CP}}) \\ &= \lambda \cdot pk_{\mathcal{CP}} = \sigma \end{aligned} \tag{2.3}$$

i pot verificar:

$$\begin{aligned} e((C' \cdot P + pk'_{\mathcal{CP}}), \sigma'_{\mathcal{CP}}) &= e((C' \cdot P + b \cdot pk_{\mathcal{CP}}), b \cdot \sigma_{\mathcal{CP}}) \\ &= e(((b \cdot H(C)) \cdot P + b \cdot pk_{\mathcal{CP}}), b \cdot \sigma_{\mathcal{CP}}) \\ &= e(b \cdot (H(C) \cdot P + sk_{\mathcal{CP}} \cdot P), b \cdot (H(C) + sk_{\mathcal{CP}})^{-1} \cdot pk_{\mathcal{U}}) \\ &= e(b \cdot P \cdot (H(C) + sk_{\mathcal{CP}}), b \cdot (H(C) + sk_{\mathcal{CP}})^{-1} \cdot pk_{\mathcal{U}}) \\ &= e(P' \cdot (H(C) + sk_{\mathcal{CP}}), b \cdot (H(C) + sk_{\mathcal{CP}})^{-1} \cdot pk_{\mathcal{U}}) \\ &= e(P' \cdot (H(C) + sk_{\mathcal{CP}}), b \cdot (H(C) + sk_{\mathcal{CP}})^{-1} \cdot sk_{\mathcal{U}} \cdot P) \\ &= e(P', b \cdot sk_{\mathcal{U}} \cdot P)^{(H(C)+sk_{\mathcal{CP}}) \cdot (H(C)+sk_{\mathcal{CP}})^{-1}} \\ &= e(P', pk'_{\mathcal{U}}) \end{aligned}$$

i també, per provar que és correcte:

$$\begin{aligned} t' \cdot P &= (s' \cdot sk'_{\mathcal{U}} + r')P \\ &= s' \cdot sk'_{\mathcal{U}} \cdot P + r'P \\ &= r'P + s' \cdot pk'_{\mathcal{U}} \\ &= R' + pk'_{\mathcal{U}} \cdot s' \end{aligned}$$

A la Figura 2.18 s'exemplifica el diagrama de l'intercanvi d'informació de forma gràfica.

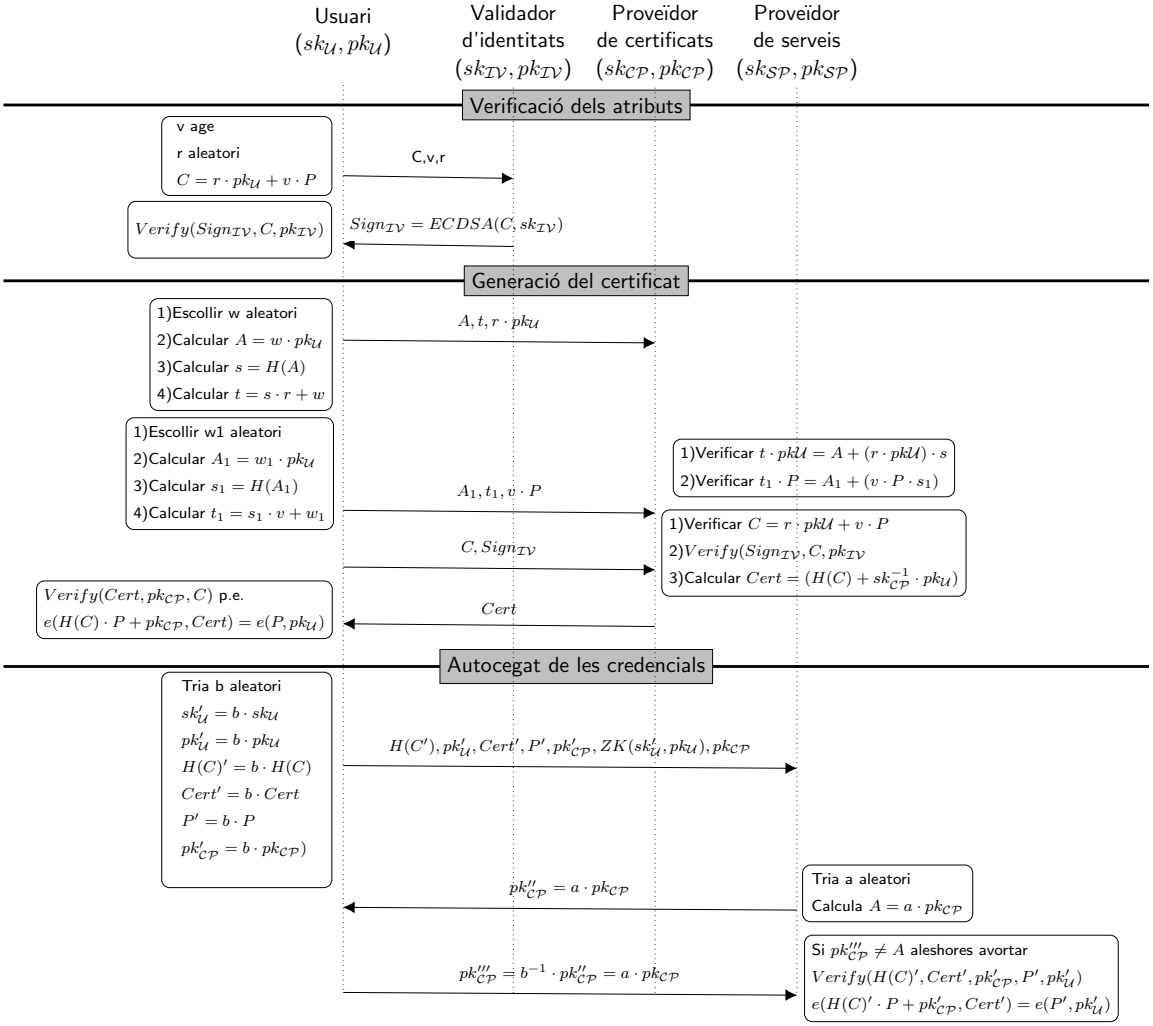

Figura 2.18: Diagrama de seqüència de l'intercanvi d'informació.

2.8 Resum

Com a resum del marc teòric estudiat, s'ha de destacar que les tecnologies basades en atributs verificables donen a l'usuari el control de les seves dades privades, i li permeten difondre-les segons la seva discreció. Cap entitat coneix dades que no li són necessàries per al seu negoci. També s'ha vist que ofereixen la possibilitat de guardar l'anonimat, i de no dependre de la comunicació amb terceres parts en el moment de la validació dels atributs verificables. Si més

no, l'anonimat ofert per aquests sistemes, no es pot aprofitar per votacions electròniques, ja que no permeten detectar la reutilització d'un atribut anònim, i per tant, no es pot assegurar la premissa: un votant, un vot. Tampoc es poden comptabilitzar quantes vegades s'utilitzen i d'aquest fet sorgeix la recerca proposada en aquest escrit.

Capítol 3

Objectius

En aquest capítol es defineix l'objectiu general de la recerca i els seus subobjectius expressats aquests darrers com a requisits tant funcionals com no funcionals.

L'objectiu general de la recerca és dissenyar i validar un protocol que permeti a un individu concret a realitzar una acció, un nombre de vegades fixat, preservant-ne el seu anonimat. A més, l'individu ha de poder exercir aquest dret lliurement, sense la necessitat de l'actuació de terceres parts de confiança.

Per la transparència, fiabilitat i descentralització s'ha optat per buscar una solució que utilitza tecnologia blockchain, més concretament una blockchain Ethereum que permet l'execució de SCs. Per satisfer aquest objectiu general, s'han definit un seguit de subobjectius, expressats com a requisits funcionals i no funcionals.

Els requisits funcionals que són les condicions que un sistema o aplicació ha de complir per poder desenvolupar-se de manera adequada, i es refereixen a les funcionalitats concretes que el sistema ha d'oferir a l'usuari, es poden fixar en quatre: només qui s'ha identificat i ha obtingut el dret pot exercir aquest dret; qui ha obtingut el dret l'ha de poder fer servir quan vulgui, sense necessitat de tercers; només podrà exercir aquest dret el nombre de vegades estipulat; i finalment, qui ha obtingut el dret està d'acord en incrementar el nombre de vegades utilitzat.

Els requisits no funcionals que són aquells que defineixen les característiques o qualitats que un sistema ha de tenir, i que estableixen com ha de funcionar el sistema més enllà de les seves accions concretes, es poden resumir en tres que són: l'exercici del dret ha de ser anònim, ningú ha de poder enllaçar l'exercici del dret a la identitat real que l'ha obtingut; els comptadors estaran allotjats sobre una blockchain Ethereum; i finalment, el cicle de vida dels comptadors estarà implementat en un SC.

El conjunt d'informació que assegura aquests objectius serà anomenat a partir d'ara "credencials". Aquestes credencials seran $\mathcal{AV}$s: un cop emeses seran anònimes i seran úniques sota certes condicions (per exemple per a una promoció donada). Es podrà enllaçar l'ús repetit de les credencials sota aquestes condicions. Un cop emeses serà l'usuari qui les custodiï. En l'obtenció de les credencials quedarà fixat el màxim nombre d'usos de la credencial, cada increment del comptador ha de contenir una prova que l'usuari està d'acord amb l'increment, ha de ser distribuït mitjançant tecnologia Blockchain i SC, i finalment ha de ser resistent davant qualsevol actor que actuï de manera maliciosa individual o en conxorxa amb d'altres.

Capítol 4

Metodologia

Aquest capítol se centra en les particularitats metodològiques que s'han seguit per assolir l'objectiu general i els requisits proposats del capítol anterior. Segons el llibre "Researching Information Systems and Computing" [53] aquesta recerca, com moltes altres en l'àmbit de Tecnologies de la Informació (TI), entra dins del model disseny i creació. En aquest model hi ha 5 fases ben diferenciades: conscienciació, suggeriment, desenvolupament, avaluació i, finalment, conclusions. En aquest capítol es desenvoluparan a l'Apartat 4.1 la conscienciació i a l'Apartat 4.2 el suggeriment, on s'analitza el problema i s'enfoca la solució, deixant les altres fases pel següent capítol.

4.1 Conscienciació

La recerca sobre les credencials anònimes enllaçables es fonamenta en una necessitat creixent de garantir la privacitat i seguretat en la gestió d'identitats digitals, especialment en contextos on la confiança en les institucions tradicionals pot ser limitada [1, 54]. En un entorn cada cop més digitalitzat i vulnerable a ciberamenaces, la possibilitat de mantenir una identitat autogestionada permet a l'individu tenir un control total sobre la seva informació personal, sense dependre de tercers per a la seva gestió i validació.

En aquest sentit, la implementació d'identitats autogestionades (Self-Sovereign Identity, SSI) utilitzant tecnologies com Blockchain i SCs s'ha consolidat com una solució robusta per garantir la privacitat i l'autenticitat de les credencials digitals. Aquest enfocament ofereix un model en què les credencials es poden verificar de manera descentralitzada i anònima, sense necessitat de revelar informació personal excessiva, reduint així els riscos associats a la recopilació centralitzada de dades.

En el cas dels comptadors d'ùs, que operen en entorns on la transparència és fonamental però la privacitat és una prioritat, la recerca se centra en com utilitzar aquestes tecnologies per establir mecanismes que permetin a l'usuari autenticar-se o validar transaccions sense exposar-se a riscos d'intrusió o manipulació. A través de l'ús de Blockchain, la informació rellevant es pot emmagatzemar de manera immutable, mentre que els SCs permeten automatitzar processos de verificació d'identitat de manera segura, garantint la integritat dels resultats sense necessitat d'un intermediari.

El repte en aquest context rau en la combinació de seguretat, privacitat i accesibilitat en un entorn hostil, on les parts involucrades poden no confiar completament entre elles. La tecnologia Blockchain, amb la seva naturalesa descentralitzada, ajuda a mitigar els riscos associats a un sistema centralitzat, mentre que els SCs proporcionen una capa de confiança automàtica que assegura que les regles preestablertes s'executin de manera fiable.

En resum, la recerca sobre les credencials anònimes enllaçables en entorns hostils com el dels comptadors d'ús implica la creació de solucions que permetin a les persones gestionar les seves pròpies identitats digitals de manera segura, privada i transparent, tot mantenint un alt nivell de confiança en el sistema a través de l'ús de la tecnologia blockchain i més concretament els SCs.

4.2 Suggeriment

En aquesta fase es proposa una manera inicial d'abordar l'objectiu. S'aplica la tècnica "divide and conquer" mitjançant un enfocament "bottom-up", on es comença amb la mínima funcionalitat i es va ampliant progressivament fins aconseguir la solució final. Així doncs, s'aniran analitzant cadascun dels requisits, tant funcionals com no funcionals, descrits en el Capítol 3, per tal d'arribar a l'objectiu general. A partir de l'objectiu principal, que és "permetre concedir a un individu identificat el dret de realitzar una acció un nombre determinat de vegades, preservant la seva privacitat", les diferents fases de l'estudi es presenten en els apartats següents, agrupant-hi un o diversos requisits.

1. Un primer requisit és "només qui s'ha identificat i ha obtingut el dret pot exercir-lo". Per tant, una primera fase consisteix en la selecció i anàlisi del mètode d'autenticació, que s'ha estudiat detalladament en el marc teòric. Seguint les conclusions exposades en l'Apartat 2.3, es conclou que l'autenticació mitjançant credencials basades en atributs és la més adequada, ja que és la possessió d'aquest atribut el que concedeix el dret d'exercir una acció. En concret, s'aplicarà la implementació analitzada a l'Apartat 2.7, que a més compleix altres requisits clau: el segon requisit, "l'exercici del dret ha de ser anònim, ningú ha de poder enllaçar l'exercici del dret a la identitat real que l'ha obtingut", i el tercer, "qui ha obtingut el dret l'ha de poder fer servir quan vulgui, sense necessitat de tercers".

2. Un altre requisit estableix que "només podrà exercir aquest dret el nombre de vegades estipulat". Després d'analitzar la solució escollida, es detecta que ofereix un nivell excessiu d'anonimat, dificultant el comptatge del nombre d'usos, ja que cada presentació de l'atribut és diferent. La solució generarà un identificador únic que, tot i permetre el comptatge de l'ús, no revela cap informació sobre la identitat real de l'individu.

3. El següent requisit a tenir en compte diu: "qui ha obtingut el dret està d'acord en incrementar el nombre de vegades utilitzat, és a dir, certa protecció contra increments il·lícits". Això implica que per dur a terme l'increment, primer s'haurà de verificar una prova del fet que el propietari de la credencial està d'acord amb l'increment.

4. Hi han dos requisits, "els comptadors estaran allotjats sobre una blockchain Ethereum" i "el cicle de vida del comptadors estarà implementat en un SC' que aboquen la recerca a buscar una implementació utilitzant la tecnologia blockchain i codificar la solució amb Solidity[1].

Per il·lustrar com funciona aquest sistema, es proposa un cas d'ús concret. Es vol crear un sistema basat en el paradigma SSI per gestionar els accessos a un servei públic, com ara un nombre determinat de viatges gratuïts en metro per a persones amb ingressos inferiors al salari mínim. En aquest escenari, hi ha tres actors principals: l'usuari $\mathcal{U}$, que és una persona identificada; l'autoritat de registre $\mathcal{AP}$, que identifica l'usuari i genera la credencial; i el proveïdor del servei $\mathcal{SP}$, que ofereix el servei en qüestió.

$\mathcal{U}$, que té certs atributs com ingressos inferiors al salari mínim, té dret a un nombre determinat d'accés a un servei concret, per exemple, 30 viatges en metro. El procés funciona de la següent manera: $\mathcal{U}$ s'identifica davant l'$\mathcal{AP}$. Aquesta verifica que l'usuari té els atributs necessaris i li proporciona una credencial digital ($\mathcal{AV}$) que acredita el seu dret a rebre el servei. L'usuari emmagatzema aquesta credencial i calcula els valors necessaris per realitzar el consum d'aquest dret les vegades que tingui dret, utilitzant una prova de coneixement nul (ZKP) per cada valor del comptador d'usos.

Cada vegada que l'usuari vol utilitzar el servei, presenta la credencial amb el valor calculat corresponent a la iteració actual al $\mathcal{SP}$. Aquest, quan rep la petició de consum del dret, verifica mitjançant el material criptogràfic rebut que qui li presenta és el propietari legítim de la credencial, que l'increment del comptador està correcte i que no s'ha exercit més de les vegades permeses. Si alguna d'aquestes condicions no es compleix, el servei es denega.

En aquest cas, les credencials o material criptogràfic són essencialment l'$\mathcal{AV}$ i la ZKP que corresponen a cada iteració. Un cop establert aquest camí a seguir, s'ha decidit descriure les parts de desenvolupament i avaluació en capítols posteriors. S'han definit clarament quatre línies de recerca, cadascuna enfocada a un conjunt de requisits específics.

Els resultats d'aquestes línies es combinaran per obtenir el resultat final que respongui a l'objectiu general del sistema. Les quatre línies de recerca estan dissenyades per abordar diferents aspectes del sistema SSI.

La primera línia se centra en la generació i verificació de credencials digitals, assegurant que les credencials són segures i verificables.

La segona línia es dedica a l'implementació de ZKP, garantint que els usuaris poden demostrar el seu dret sense revelar informació sensible.

La tercera línia aborda la gestió del comptador d'usos, assegurant que cada ús del servei està correctament registrat i verificat.

Finalment, la quarta línia es concentra en la interacció entre l'usuari, l'autoritat de registre i el proveïdor del servei, optimitzant la comunicació i la verificació per garantir una experiència d'usuari fluida.

Aquestes línies de recerca no només milloraran la funcionalitat del sistema, sinó que també

[1] URL: `https://soliditylang.org/`.

asseguraran que es compleixen els estàndards de seguretat i privacitat necessaris per a un servei públic i anònim. En combinar aquests resultats, s'obtindrà un sistema robust i eficient que permeti als usuaris accedir al servei de manera segura i verificada, sense comprometre la seva privacitat. Un cop establert el camí a seguir, s'ha decidit descriure les parts de desenvolupament i avaluació en capítols posteriors. S'han definit clarament quatre línies de recerca enfocades cadascuna a un conjunt de requisits, els resultats de les quals s'hauran de combinar per obtenir el resultat final que respongui a l'objectiu general de la tesi.

Capítol 5

Desenvolupament

Aquest capítol recull tot el desenvolupament realitzat, organitzat en diverses fases. A l'Apartat 5.1 s'aborden els requisits de "només qui s'ha identificat i ha obtingut el dret pot exercir aquest dret", "l'exercici del dret ha de ser anònim, ningú ha de poder enllaçar l'exercici del dret a la identitat real que l'ha obtingut" i "qui ha obtingut el dret l'ha de poder fer servir quan vulgui, sense necessitat de tercers", proposant i definint un sistema de credencials anònimes.

Tot i que no està associat a un requisit específic, l'Apartat 5.2 és fonamental per identificar els diferents usos d'una mateixa credencial, permetent així complir amb el requisit de "només podrà exercir aquest dret el nombre de vegades estipulat". Per aconseguir-ho, es defineix un identificador únic que facilita el control i registre d'aquests usos.

A l'Apartat 5.3 s'hi descriuen les proves de consentiment, les quals tenen com a objectiu garantir el requisit "qui ha obtingut el dret està d'acord en incrementar el nombre de vegades utilitzat, és a dir, proporcionar certa protecció contra increments il·lícits".

Finalment, l'Apartat 5.4 descriu el desenvolupament dels comptadors, els quals aborden els requisits de "només podrà exercir aquest dret el nombre de vegades estipulat", "els comptadors estaran allotjats sobre una blockchain Ethereum" i "el cicle de vida dels comptadors estarà implementat en un SC".

5.1 Protocol de credencials anònimes

El protocol de credencials anònimes parteix de la solució estudiada al protocol de referència vist a l'Apartat 2.7 que ofereix un protocol d'autenticació anònim basat en atributs verificables. En el moment que es va dur a terme la recerca, existia el codi font d'aquest article disponible a GitHub[1] el qual es va clonar per fer el desenvolupament.

En aquest apartat es fa una anàlisi de les modificacions introduïdes i necessàries per satisfer els següents suggeriments de l'Apartat 4.2: "només qui s'ha identificat i ha obtingut el dret pot exercir aquest dret", "l'exercici del dret ha de ser anònim, ningú ha de poder enllaçar l'exercici del dret a la identitat real que l'ha obtingut" i "qui ha obtingut el dret l'ha de poder fer servir quan vulgui, sense necessitat de tercers".

[1] URL: `https://github.com/odib/privacy-preserving-credential-scheme-over-blockchain.git`.

5.1.1 Arquitectura

En aquesta recerca, l'objectiu principal és preservar l'anonimat de l'usuari en un entorn hostil, on no es confia en cap actor per mantenir la privacitat de les credencials de l'individu. En el protocol de referència, l'actor que té la capacitat d'enllaçar les credencials anònimes amb la identitat real és l'$\mathcal{IV}$. Aquest escenari introdueix un risc important, ja que una possible conxorxa entre l'$\mathcal{IV}$ i el $\mathcal{SP}$ permetria vulnerar la privacitat de l'usuari, permetent vincular la seva identitat real amb les seves credencials anònimes.

Per tal d'evitar aquesta amenaça, s'ha decidit eliminar l'$\mathcal{IV}$ del sistema, la qual cosa elimina la possibilitat d'una conxorxa entre aquest actor i el $\mathcal{SP}$. Aquesta modificació redueix el nombre d'actors implicats en el procés de validació de les credencials, passant de quatre a tres: $\mathcal{U}$, $\mathcal{AP}$, i $\mathcal{SP}$. D'aquesta manera, s'assegura que no existeix cap actor intermediari que pugui vincular les credencials anònimes amb la identitat real de l'usuari, preservant així l'anonimat en tot moment, fins i tot en entorns hostils.

5.1.2 Emissió de credencials

És en aquest punt del desenvolupament que s'introdueix el concepte d'àmbit S, un valor alfanumèric arbitrari que serveix per proporcionar un identificador únic en un determinat context. A partir d'aquesta introducció, es defineix un valor de compromís o commit $C = H(S)$, on H representa una funció de hash criptogràfica. Aquest valor de commit és fix i, per tant, elimina la necessitat de la fase de commit present en el protocol original. En el protocol de referència, la fase de commit permetia assegurar que les credencials emeses no es podrien modificar posteriorment, però en el nou enfocament, el valor C derivat de l'àmbit S ja proporciona aquesta garantia, fent innecessària aquesta etapa addicional.

En eliminar l'$\mathcal{IV}$, s'han fusionat dos processos previs del protocol original: la signatura del commit i l'emissió de la credencial, convertint-los en un únic pas. Aquesta simplificació s'ha acompanyat d'un canvi en el nom de l'actor implicat en el procés de validació, passant del $\mathcal{CP}$ a l'$\mathcal{AP}$, que ara s'encarrega de validar les credencials en un sol pas. En aquest nou flux, el valor de commit es basa directament en el hash de l'àmbit S, garantint que les credencials emeses estan associades de manera única i segura a un àmbit específic. Així, l'emissió de les credencials es simplifica i es fa més segura, sense comprometre la privacitat ni l'anonimat de l'usuari.

Per tant, la nova metodologia per l'emissió de les credencials es resumeix en el següent procés:

1. $\mathcal{U}$ demana la credencial a $\mathcal{AP}$.
2. $\mathcal{AP}$ verifica la identitat i els possibles requisits per poder atorgar les credencials dins l'àmbit S demanat per $\mathcal{U}$.
3. $\mathcal{AP}$ genera la signatura amb l'esquema de signatura curta de ZSS per aparellaments bilineals
$$\sigma_{\mathcal{AP}} = (H(S) + sk_{\mathcal{AP}})^{-1} \cdot pk_{\mathcal{U}}$$
4. $\mathcal{AP}$ envia $\sigma_{\mathcal{AP}}, pk_{\mathcal{AP}}$ a $\mathcal{U}$.

5. $\mathcal{U}$ verifica la signatura rebuda:

$$\begin{aligned} e(H(S)\cdot P + pk_{\mathcal{AP}}, \sigma_{\mathcal{AP}}) &= e((H(S)\cdot P + sk_{\mathcal{AP}}\cdot P), (H(S)\cdot sk_{\mathcal{AP}})^{-1}\cdot pk_{\mathcal{U}}) \\ &= e((H(S)+sk_{\mathcal{AP}})\cdot P, (H(S)+sk_{\mathcal{AP}})^{-1}\cdot pk_{\mathcal{U}}) \\ &= e(P, pk_{\mathcal{U}})^{(H(S)+sk_{\mathcal{AP}})\cdot(H(S)+sk_{\mathcal{AP}})^{-1}} \\ &= e(P, pk_{\mathcal{U}}) \end{aligned} \tag{5.1}$$

Si la verificació de la signatura és correcta, $\mathcal{U}$ emmagatzema la credencial $\sigma_{\mathcal{AP}}$ per a futurs usos. $\sigma_{\mathcal{AP}}$ és la credencial que dona accés a $\mathcal{U}$ per l'àmbit S.

5.1.3 Presentació de credencials

El protocol de base finalitza amb una interacció entre $\mathcal{U}$ i $\mathcal{SP}$, tal com es descriu en el procés original. Aquesta interacció permet que l'usuari pugui exercir els drets associats a la seva credencial, garantint que només qui ha obtingut el dret pugui fer-ne ús.

No obstant això, per millorar la privadesa de les activitats de l'usuari dins de la blockchain, s'afegeix una característica addicional basada en un esquema d'autocegat. L'autocegat té com a objectiu protegir la identitat de l'usuari en evitar que les seves activitats siguin traçables a través de la blockchain, mantenint l'anonimat fins i tot en un entorn descentralitzat i públic com aquest. Per aconseguir-ho, es fa ús de la signatura curta modificada ZSS, juntament amb l'esquema d'autocegat de Verheul [54], que permet ofuscar les claus utilitzades en el procés, en relació amb les entregades pel $\mathcal{CP}$.

Aquestes credencials cegades continuen sent verificables i mantenen la signatura de l'$\mathcal{AP}$, de manera que, tot i que les claus s'han cedit a l'esquema d'autocegat, la integritat i l'autenticitat de les credencials romanen intactes. Això garanteix que l'usuari pot demostrar la seva identitat de manera anònima i segura, sense que ningú pugui enllaçar la seva activitat a la identitat real, tot mantenint la confiança i seguretat que exigeix el sistema.

Els passos de la solució proposada segueixen essent essencialment els mateixos que els del protocol base analitzat a l'Apartat 2.7, amb les úniques diferències que consisteixen en el canvi de l'actor $\mathcal{CP}$ per $\mathcal{AP}$ i l'adopció del valor S en lloc de C, com a valor de compromís derivat de l'àmbit S. Aquests passos són:

1. $\mathcal{U}$ té $\sigma_{\mathcal{AP}}$, C, $sk_{\mathcal{U}}$, $pk_{\mathcal{U}}$.
2. $\mathcal{U}$ tria $b \in_R \mathbb{Z}_q^*$ com a factor d'ofuscació.
3. $\mathcal{U}$ calcula per una banda:
   - $sk'_{\mathcal{U}} = b\cdot sk_{\mathcal{U}}$, $pk'_{\mathcal{U}} = b\cdot sk_{\mathcal{U}}\cdot P$
   - $\sigma'_{\mathcal{AP}} = b\cdot \sigma_{\mathcal{AP}}$
   - $P' = b\cdot P$
   - $pk'_{\mathcal{AP}} = b\cdot pk_{\mathcal{AP}}$
   - $C' = b\cdot H(C)$

4. $\mathcal{U}$ també calcula la prova de coneixement nul NI-Schnorr:
   - escull $r' \in_R \mathbb{Z}_q^*$
   - calcula:
     - $R' = r' \cdot P$
     - $s' = H(R')$
     - $t' = s' \cdot sk'_{\mathcal{U}} + r'$
5. $\mathcal{U}$ envia (R', s', t') per la prova NI-Schnorr i $\sigma'_{\mathcal{AP}}$, $pk'_{\mathcal{U}}$, $pk'_{\mathcal{AP}}$, P', C' per la validació de les credencials.
6. $\mathcal{SP}$ tria $\lambda \in_R \mathbb{Z}_q^*$ i calcula:
   - $\sigma = \lambda \cdot pk_{\mathcal{AP}}$
   - $pk''_{\mathcal{AP}} = \lambda \cdot pk'_{\mathcal{AP}}$
7. $\mathcal{SP}$ envia $pk''_{\mathcal{AP}}$
8. $\mathcal{U}$ calcula $pk'''_{\mathcal{AP}} = b^{-1} \cdot pk''_{\mathcal{AP}}$
9. $\mathcal{U}$ envia $pk'''_{\mathcal{AP}}$
10. $\mathcal{CP}$ si $\sigma \equiv pk'''_{\mathcal{AP}} \Rightarrow pk'_{\mathcal{AP}}$ és correcte:

$$\begin{aligned} pk'''_{\mathcal{AP}} &= b^{-1} \cdot pk''_{\mathcal{AP}} \\ &= b^{-1} \cdot \lambda \cdot pk'_{\mathcal{AP}} \\ &= b^{-1} \cdot \lambda \cdot (b \cdot pk_{\mathcal{AP}}) \\ &= \lambda \cdot pk_{\mathcal{AP}} = \sigma \end{aligned} \tag{5.2}$$

i pot verificar:

$$\begin{aligned} e((C' \cdot P + pk'_{\mathcal{AP}}), \sigma'_{\mathcal{AP}}) &= e((C' \cdot P + b \cdot pk_{\mathcal{AP}}), b \cdot \sigma_{\mathcal{AP}}) \\ &= e(((b \cdot H(C)) \cdot P + b \cdot pk_{\mathcal{AP}}), b \cdot \sigma_{\mathcal{AP}}) \\ &= e(b \cdot (H(C) \cdot P + sk_{\mathcal{AP}} \cdot P), b \cdot (H(C) + sk_{\mathcal{AP}})^{-1} \cdot pk_{\mathcal{U}}) \\ &= e(b \cdot P \cdot (H(C) + sk_{\mathcal{AP}}), b \cdot (H(C) + sk_{\mathcal{AP}})^{-1} \cdot pk_{\mathcal{U}}) \\ &= e(P' \cdot (H(C) + sk_{\mathcal{AP}}), b \cdot (H(C) + sk_{\mathcal{AP}})^{-1} \cdot pk_{\mathcal{U}}) \\ &= e(P' \cdot (H(C) + sk_{\mathcal{AP}}), b \cdot (H(C) + sk_{\mathcal{AP}})^{-1} \cdot sk_{\mathcal{U}} \cdot P) \\ &= e(P', b \cdot sk_{\mathcal{U}} \cdot P)^{(H(C)+sk_{\mathcal{AP}}) \cdot (H(C)+sk_{\mathcal{AP}})^{-1}} \\ &= e(P', pk'_{\mathcal{U}}) \end{aligned}$$

i també, per provar que és correcte:

$$\begin{aligned} t' \cdot P &= (s' \cdot sk'_{\mathcal{U}} + r')P \\ &= s' \cdot sk'_{\mathcal{U}} \cdot P + r'P \\ &= r'P + s' \cdot pk'_{\mathcal{U}} \\ &= R' + pk'_{\mathcal{U}} \cdot s' \end{aligned}$$

Aquestes modificacions es mantenen coherents amb l'objectiu de millorar la privadesa i l'anonimat de l'usuari, tot preservant la funcionalitat i la seguretat del protocol original.

5.2 L'identificador únic d'usuari

El protocol presentat a l'Apartat 5.1, però, no permet detectar la reutilització de la credencial, és massa anònim. Per poder realitzar aquesta detecció, es defineix un identificador únic, basat en el càlcul de la signatura curta ZSS. És una autosignatura que es pot comprovar amb la credencial ofuscada del protocol de credencials anònimes vist a l'Apartat 5.1, és generada per l'usuari, ningú més pot associar-la a la identitat real i, al ser única i no haver-se d'ofuscar, permet la detecció de la seva reutilització. La idea és construir un identificador dependent de la clau privada $sk_{\mathcal{U}}$, de l'àmbit i impossible d'enllaçar amb la identitat real. L'opció escollida és una modificació de l'esquema de signatura curta de ZSS per aparellaments bilineals que permetrà la seva verificació de la mateixa manera que les credencials anònimes vistes a l'Apartat 5.1.

Si es defineix $id_{\mathcal{U}} = X \cdot pk_{\mathcal{U}}$

$$\begin{aligned} e(C' + pk'_{\mathcal{U}}, H(S) \cdot id_{\mathcal{U}}) &= e(b \cdot H(S) \cdot P + pk'_{\mathcal{U}}, H(S) \cdot X \cdot pk_{\mathcal{U}}) \\ &= e(b \cdot H(S) \cdot P + b \cdot sk_{\mathcal{U}} \cdot P, H(S) \cdot X \cdot pk_{\mathcal{U}}) \\ &= e((H(S) + sk_{\mathcal{U}}) \cdot b \cdot P, H(S) \cdot X \cdot pk_{\mathcal{U}}) \\ &= e((H(S) + sk_{\mathcal{U}}) \cdot P, H(S) \cdot X \cdot pk_{\mathcal{U}})^b \\ &= e(P, b \cdot pk_{\mathcal{U}})^{(H(S)+sk_{\mathcal{U}}) \cdot H(S) \cdot X} \\ &= e(P, b \cdot pk_{\mathcal{U}})^{(H(S)+sk_{\mathcal{U}}) \cdot H(S) \cdot (H(S)+sk_{\mathcal{U}})^{-1} \cdot H(S)^{-1}} \\ &= e(P, pk'_{\mathcal{U}}) \end{aligned}$$

per tant la signatura modificada que representa el nostre identificador global serà :

$$id_{\mathcal{U}} = X \cdot pk_{\mathcal{U}} = H(S)^{-1}(sk_{\mathcal{U}} + H(S))^{-1} \cdot pk_{\mathcal{U}}$$

Ara sí, $\mathcal{SP}$ pot verificar també l'identificador:

$$\begin{aligned} e(C' + pk'_{\mathcal{U}}, H(S) \cdot id_{\mathcal{U}}) &= e(b \cdot H(S) \cdot P + b \cdot pk_{\mathcal{U}}, H(S) \cdot H(S)^{-1} \cdot (sk_{\mathcal{U}} + H(S))^{-1} \cdot pk_{\mathcal{U}}) \\ &= e(b \cdot (H(S) \cdot P + sk_{\mathcal{U}} \cdot P), (H(S) + sk_{\mathcal{U}})^{-1} \cdot pk_{\mathcal{U}}) \\ &= e(P \cdot (H(S) + sk_{\mathcal{U}}), (H(S) + sk_{\mathcal{U}})^{-1} \cdot pk_{\mathcal{U}})^{b} \\ &= e(P, b \cdot pk_{\mathcal{U}})^{(H(S)+sk_{\mathcal{U}}) \cdot (H(S)+sk_{\mathcal{U}})^{-1}} \\ &= e(P, pk'_{\mathcal{U}}) \end{aligned} \tag{5.3}$$

5.3 Les proves de consentiment

L'objectiu principal és introduir el consentiment exprés de l'usuari en l'ús de les credencials, amb la finalitat de prevenir atacs de repetició. Aquest procés implica la integració de dos valors, n i i, on n representa el nombre màxim d'usos i i el valor d'increment, en el càlcul del NI-ZKP. A continuació, es presenten tres opcions per incorporar aquests valors, basades en el càlcul original de la prova, com es descriu a [55]:

$$\begin{aligned} R' &= r' \cdot P \\ t &= H(id_{\mathcal{U}}) \cdot sk'_{\mathcal{U}} + r' \end{aligned} \tag{5.4}$$

Opció 1: Multiplicació directa de t per i

La primera opció per introduir el valor d'increment i és multiplicar el valor t per i. Així, la Fórmula 5.4 es modifica de la següent manera:

$$t_i = i \cdot t = i \cdot (H(id_{\mathcal{U}}) \cdot sk'_{\mathcal{U}} + r'), \quad \forall i \in (0, 1, .., n)$$

No obstant això, aquesta opció presenta una vulnerabilitat: si es coneix el valor de t_1, és relativament fàcil forjar els següents valors de t $(t_2, t_3, \dots)$ mitjançant la Fórmula $t_2 = t_1 + t_1$, i així successivament. Això permetria la creació de valors falsos de t.

Opció 2: Multiplicació de r' per i

La segona opció consisteix a multiplicar només el terme que conté r' en la Fórmula de t. La nova expressió seria:

$$t_i = H(id_{\mathcal{U}}) \cdot sk'_{\mathcal{U}} + i \cdot r', \quad \forall i \in (0, 1, .., n)$$

En aquest cas, la relació entre t_{i+1} i t_i seria:

$$\begin{aligned} t_{i+1} &= (i+2) \cdot R' + pk'_{\mathcal{U}} \cdot H(id_{\mathcal{U}}) \\ &= R' + (i+1) \cdot R' + pk'_{\mathcal{U}} \cdot H(id_{\mathcal{U}}) \\ &= R' + t_i, \quad \forall i \in [0..n] \end{aligned}$$

Aquest enfocament presenta el problema que qualsevol persona que conegui R' (públic) podria afegir-lo a t_i, i així generar un t_{i+1} fals. Això exposa el sistema a un atac de forjament de t.

Opció 3: Multiplicació de la clau privada per i

La tercera i darrera opció consisteix a multiplicar el terme que conté la clau privada en el càlcul de t. Aquesta opció s'expressa com:

$$t_i = r' + i \cdot sk'_{\mathcal{U}} \cdot H(id_{\mathcal{U}})$$

Amb aquest mètode, ningú pot forjar el valor de t_i sense conèixer la clau privada de l'usuari. Aquesta és l'estratègia seleccionada per a aquest estudi, ja que ofereix una seguretat robusta contra atacs de forjament.

Evitar la multiplicació per zero

Una vegada escollida l'opció per introduir els valors n i i en el ZKP, es presenta un problema en el cas que i sigui igual a zero. En aquest cas, la multiplicació de la clau privada per zero resultaria en un valor nul, la qual cosa no és desitjable. Per evitar aquesta situació, es modifica el factor de multiplicació en el càlcul de t_i a $i + 1$, evitant així la multiplicació per zero.

Demostració de la robustesa del protocol

El protocol actual es basa en la robustesa del ZKP, que ha de ser validada mitjançant demostracions matemàtiques. A partir de la Fórmula 5.4, la demostració per als valors públics i, $id_{\mathcal{U}}$ i $sk'_{\mathcal{U}}$ es calcula com:

$$\begin{aligned} R' &= r' \cdot P \\ t_i &= (i+1) \cdot H(id_{\mathcal{U}}) \cdot sk'_{\mathcal{U}} + r' \end{aligned}$$

Aquesta expressió mostra que, mentre els valors $sk'_{\mathcal{U}}$ i r' siguin privats per a l'usuari $\mathcal{U}$, ningú més podrà forjar un ZKP. Aquest resultat es basa en la dificultat del problema de logarithmes discrets (DLP) en grups additius, tal com es descriu en el problema 1 de l'apartat 2.2.3.

Amenaces i seguretat del protocol

Els principals riscos associats al protocol són les següents amenaces:

- L'ús d'un identificador únic d'usuari ($id_{\mathcal{U}}$) arbitrari en què un $\mathcal{U}$ maliciós intenta forjar un comptador sense haver completat la fase d'autenticació.
- Comportament maliciós de $\mathcal{SP}$ durant la fase de creació del comptador, intentant generar un comptador amb un valor d'ús màxim incorrecte.
- Atac de repetició realitzat per $\mathcal{SP}$.

- Comportament maliciós de $\mathcal{SP}$ que intenta forjar un valor de t_i no vàlid.

No obstant això, gràcies a la robustesa del ZKP, aquestes amenaces es poden mitigar de manera eficient.

Càlcul de les proves de consentiment

En resum, les proves de consentiment (R, t_i) per cada i es calculen com segueix:

$$\begin{aligned} R &= r' \cdot P \\ t_i &= (i+1) \cdot H(id_{\mathcal{U}}) \cdot sk'_{\mathcal{U}} + r' \end{aligned}$$

Aquesta Fórmula assegura que les proves siguin vàlides i no puguin ser falsificades sense conèixer la clau privada de l'usuari.

Comprovació de la prova a la i-èsima iteració

La comprovació de la prova a la i-èsima iteració es realitza tenint en compte els paràmetres públics $(R, t_i, pk'_{\mathcal{U}}, id_{\mathcal{U}}, P)$, i es calcula com segueix:

$$\begin{aligned} t_i \cdot P &= ((i+1) \cdot H(id_{\mathcal{U}}) \cdot sk'_{\mathcal{U}} + r') \cdot P \\ &= (i+1) \cdot H(id_{\mathcal{U}}) \cdot sk'_{\mathcal{U}} \cdot P + r' \cdot P \\ &= (i+1) \cdot H(id_{\mathcal{U}}) \cdot pk'_{\mathcal{U}} + R \end{aligned} \tag{5.5}$$

5.4 Els comptadors

La part més senzilla però la que dona més valor afegit a la solució són els comptadors. Està pensat com un SC desplegat en una blockchain Ethereum per transparència i per la característica de descentralització. Aquest SC s'invoca cada cop que es vol usar la credencial amb els paràmetres adients i, si la crida acaba correctament, es pot donar l'accés i ja queda comptabilitzat.

Així, cada comptador és un registre de dades emmagatzemat en la cadena amb un seguit de dades:

- Un identificador associat a unes credencials anònimes amb una clau pública ofuscada per tal de conservar l'anonimat, s'agafa com a identificador el hash de l'$id_{\mathcal{U}}$.
- El màxim valor al que pot arribar el comptador i a partir del qual ja queda exhaurit.
- El nombre d'usos utilitzats per portar el control
- La clau pública ofuscada de l'usuari propietari

A banda de l'estructura de dades necessària per emmagatzemar les dades, l'SC ha de proporcionar mètodes per interactuar amb el comptador, aquests són: el mètode *Create* i el mètode *Consume* que es detallen a continuació.

5.4.1 Mètode Create

Les credencials venen donades pel conjunt de paràmetres $\sigma'_{\mathcal{AP}}$, $pk'_{\mathcal{U}}$, $pk'_{\mathcal{AP}}$, P', C' de l'Apartat 5.1 més l'$id_{\mathcal{U}}$ de l'Apartat 5.2. L'estructura de dades que conté la informació necessària per fer funcionar els comptadors s'ha d'inicialitzar abans de poder utilitzar-se; per tant, l'SC té el mètode Create que fa aquesta funció. Té com a paràmetres d'entrada les credencials anònimes, el nombre màxim d'usos de les credencials i la NI-ZKP, de l'Apartat 5.3, que prova l'acceptació del nombre màxim d'usos per part del que posseeix les credencials. Serà el hash de l'$id_{\mathcal{U}}$, que és a l'hora únic i anònim, l'identificador del comptador d'usos de les credencials. Així, l'SC té un magatzem de comptadors indexat amb el hash de l'$id_{\mathcal{U}}$.

Aquest mètode és el responsable de la validació de les credencials abans de la creació del comptador, per tant és l'encarregat d'efectuar les diferents comprovacions:

1. Comprovar que la clau pública cegada d'$\mathcal{AP}$ de les credencials correspon a la clau pública coneguda d'$\mathcal{AP}$, implica validar la fórmula :

$$e(pk'_{\mathcal{AP}}, P) \stackrel{?}{=} e(pk_{\mathcal{AP}}, P') \tag{5.6}$$

2. Comprovar la credencial ofuscada $\sigma'_{\mathcal{AP}}$, cosa que valida la Fórmula 5.1.

3. Comprovar que l'$id_{\mathcal{U}}$ estigui associat a les claus de les credencials, mitjançant la Fórmula 5.3.

4. I finalment, comprovar si l'NI-ZKP és bona amb la clau pública ofuscada de les credencials i el número màxim d'usos amb la Fórmula 5.5 fixant i a aquest número màxim.

Només que una de les comprovacions falli, es reverteix la transacció no creant doncs el comptador. Si la transacció acaba satisfactòriament, l'SC inicialitza i emmagatzema la informació relativa al comptador a la cadena i l'usuari ja pot utilitzar les credencials amb el comptador, fins a que l'exhaureixi.

5.4.2 Mètode Consume

A l'i-èsima iteració les credencials són (R, t_i), $pk'_{\mathcal{U}}$, $id_{\mathcal{U}}$. Les comprovacions abans de donar per bo l'accés són menys que en el cas de la creació. En aquest cas només cal verificar quatre condicions:

1. Que existeix un registre que conté les dades del comptador amb aquest $id_{\mathcal{U}}$, el que assegura que ha passat les validacions del mètode Create.

2. Cal validar la prova de consentiment sobre l'increment, Fórmula 5.5

3. I finalment que el nombre màxim d'usos no s'hagi exhaurit, és a dir, que el valor gravat a la cadena sigui inferior al màxim.

Només si totes les validacions són correctes, l'SC incrementarà el valor actual del comptador a la cadena, no farà revert de la transacció i l'$\mathcal{SP}$ pot oferir el servei.

5.4.3 Pseudo-codi

Aquesta secció proporciona el pseudocodi de l'SC implementat. Cada fórmula de verificació de les seccions anteriors té la seva pròpia funció *TESTXX()*. Cadascun retorna un valor booleà com a resultat de la seva avaluació. S'ha de tenir en compte que la variable *params* és una *struct* amb els paràmetres necessaris a la funció. La funció *assert* reverteix l'execució del SC si l'avaluació és **false**.

```
Contract counter {
    storage bytes pkAP
    struct record{
      integer max_value
      integer current_value
      bytes blinded_public_key_user
    }
    storage records array of record

    constructor counter (params)
    {
      pkAP = params.pkAP
    }

    function Create(params) {
      assert(not exist records[hash(params.idu)])
      assert(TESTXX)
      ....
      assert(TESTXX)
      create record r with:
        max_value = params.max_value
        current_value = 0
        blinded_public_key_user = params.blinded_public_key_user
      store records[hash(params.idu)] = r
    }

    function Consume(params) {
      assert(exist records[hash(params.idu)])
      r = records[hash(params.idu)]
      assert(TEST18(params,r.blinded_public_key_user))
      assert(r.current_value < r.max_value)
      r.current_value++
      store records[hash(params.idu)] = r
    }
}
```

Llistat 5.1: pseudo-codi del Smart Contract

Capítol 6

Resultats

Aquest capítol presenta els resultats finals de la recerca, proporcionant una visió detallada de les proves i avaluacions realitzades. Comença amb la descripció de la solució final que assoleix els objectius generals de l'estudi. A continuació, s'analitza la seguretat del protocol implementat, seguit d'una validació exhaustiva per assegurar-ne la seva efectivitat.

El capítol també cobreix diverses implementacions realitzades i detalla els experiments i mesures efectuats sobre el protocol. Es descriuen les validacions del protocol desplegat en una cadena privada, així com els experiments i mesures específics realitzats en aquest entorn. A més, s'inclouen proves de rendiment per identificar les limitacions de la solució.

Finalment, es fa un resum complet del capítol, sintetitzant totes les informacions presentades i destacant els punts clau dels resultats obtinguts.

6.1 Credencials anònimes enllaçables basades en atributs verificables amb comptadors d'ús

Per satisfer l'objectiu general del Capítol 3 s'ha dut a terme el disseny d'un protocol d'autenticació anònim basat en atributs verificables, i que a més d'anònim fos enllaçable en un àmbit donat, és a dir que repetits usos d'aquestes credencials anònimes es poguessin detectar i comptar dins d'aquest àmbit. Així, s'ha partit del protocol presentat a l'Apartat 2.7 que ofereix anonimat, i se li ha afegit el càlcul d'un identificador únic per l'àmbit escollit. Tant el protocol original, com el derivat es fonamenten en Elliptic Curve Criptography (ECC) i ZKP. Seguidament es fa una descripció detallada d'aquest protocol.

6.1.1 Visió general

Entren en joc quatre actors en el protocol: ($\mathcal{U}$) l'usuari que vol utilitzar les credencials, ($\mathcal{AP}$) el proveïdor d'atributs que proporciona les credencials, ($\mathcal{SP}$) el proveïdor de serveis que dona el servei a qui demostri que posseeix el permís que li atorguen les credencials anònimes i, com no, la Blockchain que serà l'encarregada d'assegurar el protocol.

Aquest protocol consisteix en dos subprotocols: (i) l'obtenció de les credencials i (ii) l'ús de les credencials. Les credencials consisteixen en una tupla de nombres que l'usuari obté de

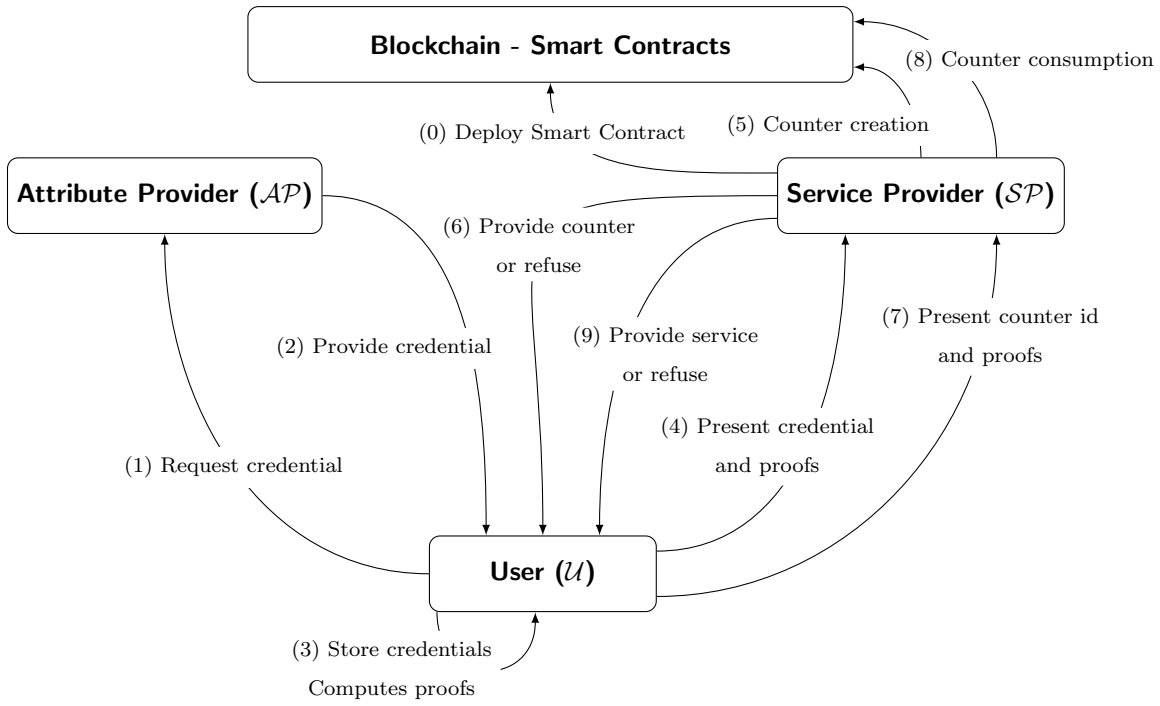

Figura 6.1: Visió general dels intercanvis d'informació entre actors.

l'$\mathcal{AP}$ després de l'execució de (i) i que, a més, ha de custodiar. Després, un cop ja té les credencials, pot utilitzar-les quan ell vulgui mitjançant (ii) davant de l'$\mathcal{SP}$. Per aquesta raó, es pot considerar que és un sistema autogestionat o Self Sovereign.

El resultat de l'execució del protocol (i) són dos nombres que només es poden utilitzar conjuntament. Un és l'identificador únic associat a l'àmbit i l'altre és la prova que aquest certificat ha estat generat per l'$\mathcal{AP}$. Ambdós són anònims, és a dir, ningú no pot associar-los a una identitat real (o només pot fer-ho amb una probabilitat molt petita), i alhora asseguren que han estat generats per l'$\mathcal{AP}$ que pot assegurar la legitimitat d'aquests, és a dir, que la identitat que l'ha demanat és correcta i té dret a poder exercir aquest dret. Finalment, quan s'utilitzin hi haurà un tercer nombre, calculat a cada cop per $\mathcal{U}$, que és una ZKP que permet assegurar que qui l'utilitza és el propietari de la credencial i que està d'acord en incrementar el seu comptador d'usos.

Una explicació més detallada i una representació gràfica d'aquest mecanisme es pot veure a la Figura 6.1 on:

(0) $\mathcal{SP}$ desplega l'SC a la Blockchain.

(1) $\mathcal{U}$ comença el protocol demanant a $\mathcal{AP}$ unes credencials per un àmbit donat S.

(2) Després d'haver verificat la identitat de l'usuari, $\mathcal{AP}$ proporciona la credencial a l'usuari. Aquesta credencial només serà vàlida a l'àmbit S pel que s'ha fet la petició.

(3) $\mathcal{U}$ guarda les credencials.

(4) $\mathcal{U}$ presenta les credencials, i a ZKP que prova la possessió de la clau privada i l'acord amb el nombre màxim d'usos a $\mathcal{SP}$ per la creació i inicialització del comptador d'usos a

la Blockchain.

(5) $\mathcal{SP}$ realitza una crida de creació de comptador al SC. Aquest realitza les comprovacions necessàries i si acaba sense errors $\mathcal{SP}$ proporciona dit comptador a $\mathcal{U}$.

(6) $\mathcal{SP}$ comunica el resultat de l'execució de SC a $\mathcal{U}$.

(7) si la creació ha estat correcte, $\mathcal{U}$ per utilitzar les credencials, ha de presenta a $\mathcal{SP}$ el seu $id_{\mathcal{U}}$, juntament amb una ZKP calculada amb la clau privada i el següent valor del comptador.

(8) $\mathcal{SP}$ realitza una crida de consum del comptador al SC. Aquest realitza les comprovacions necessàries.

(9) si la crida anterior acaba sense errors $\mathcal{SP}$ proporciona el servei a $\mathcal{U}$.

Els passos 7, 8 i 9 es van repetint fins que el nombre d'usos sobrepassa els marcats a l'inici. Aleshores l'SC fa un revert i $\mathcal{SP}$ ha de deixar d'oferir el servei.

6.1.2 Generació de les claus

Sigui $\mathbb{F}_q$ un grup cíclic d'ordre q, on q és primer, amb un punt generador $P \in E(\mathbb{F}_q)$. cada actor escull un número aleatori $sk \in_R \mathbb{F}_q$ que serà la seva clau privada, i calcula el producte escalar d'aquest amb el generador, obtenint així la seva clau pública pk. D'aquesta manera, el sistema tindrà tres parells de claus: $(sk_{\mathcal{U}}, pk_{\mathcal{U}})$, $(sk_{\mathcal{AP}}, pk_{\mathcal{AP}})$, $(sk_{\mathcal{SP}}, pk_{\mathcal{SP}})$. Aquestes claus pertanyen a $\mathcal{U}$, $\mathcal{AP}$ i $\mathcal{SP}$ respectivament.

6.1.3 Generació de les credencials ($\mathcal{U} \Longleftrightarrow \mathcal{AP}$)

Quan l'usuari $\mathcal{U}$ demana les credencials a l'$\mathcal{AP}$, aquest darrer, abans de res, ha de verificar la identitat d'$\mathcal{U}$ per qualsevol mitjà necessari que inclou, per exemple, documents físics o verificació cara a cara. Llavors el registrarà com a membre autoritzat per l'àmbit S i li proporcionarà les credencials.

L'intercanvi de missatges entre els dos actors és (veure Figura 6.2):

1. $\mathcal{U}$ demana unes credencials per l'àmbit S.
2. $\mathcal{AP}$ verifica la identitat i els possibles requisits per poder atorgar les credencials dins l'àmbit S demanat per l'usuari.
3. $\mathcal{AP}$ genera la signatura amb l'esquema de signatura curta de ZSS per aparellaments bilineals
$$\sigma_{\mathcal{AP}} = (H(S) + sk_{\mathcal{AP}})^{-1} \cdot pk_{\mathcal{U}} \tag{6.1}$$
4. $\mathcal{AP}$ envia $\sigma_{\mathcal{AP}}, pk_{\mathcal{AP}}$ a $\mathcal{U}$.

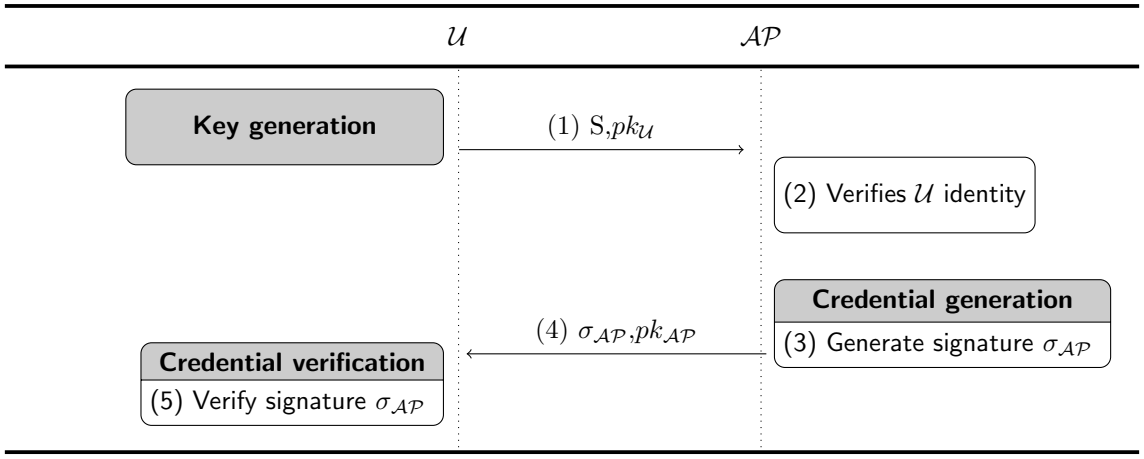

Figura 6.2: Intercanvi de missatges entre $\mathcal{U}$ i $\mathcal{AP}$ durant la fase de generació de credencials.

5. $\mathcal{U}$ verifica la signatura rebuda:

$$\begin{aligned} e(H(S)\cdot P + pk_{\mathcal{AP}}, \sigma_{\mathcal{AP}}) &= e((H(S)\cdot P + sk_{\mathcal{AP}}\cdot P), (H(S)\cdot sk_{\mathcal{AP}})^{-1}\cdot pk_{\mathcal{U}}) \\ &= e((H(S)+sk_{\mathcal{AP}})\cdot P, (H(S)+sk_{\mathcal{AP}})^{-1}\cdot pk_{\mathcal{U}}) \\ &= e(P, pk_{\mathcal{U}})^{(H(S)+sk_{\mathcal{AP}})\cdot(H(S)+sk_{\mathcal{AP}})^{-1}} \\ &= e(P, pk_{\mathcal{U}}) \end{aligned}$$

Si la verificació de la signatura és correcta, $\mathcal{U}$ emmagatzema la credencial $\sigma_{\mathcal{AP}}$ per a futurs usos. $\sigma_{\mathcal{AP}}$ és la credencial que dona accés a $\mathcal{U}$ per l'àmbit S.

6.1.4 Creació del comptador ($\mathcal{U} \Longrightarrow \mathcal{SP}$)

En aquest protocol, $\mathcal{U}$ utilitza un esquema d'autoofuscat per preservar la seva privacitat. L'intercanvi de missatges es mostra a la Figura 6.3.

$\mathcal{U}$, per calcular el seu identificador únic, genera la signatura amb l'esquema de signatura ZSS per aparellaments bilineals

$$id_{\mathcal{U}} = H(S)^{-1}(sk_{\mathcal{U}} + H(S))^{-1}\cdot pk_{\mathcal{U}} \tag{6.2}$$

Les credencials així emmascarades segueixen essent verificables i mantenen la signatura d'$\mathcal{AP}$. Els passos per obtenir aquests valors són:

1. $\mathcal{U}$ coneix $\sigma_{\mathcal{AP}}$, $H(S)$, $sk_{\mathcal{U}}$, $pk_{\mathcal{U}}$ i $id_{\mathcal{U}}$.

2. $\mathcal{U}$ escull $b \in_R \mathbb{F}_q^*$ com a factor d'ofuscació.

3. $\mathcal{U}$ calcula:

$$\begin{aligned} sk'_{\mathcal{U}} &= b \cdot sk_{\mathcal{U}} \\ pk'_{\mathcal{U}} &= sk'_{\mathcal{U}} \cdot P \\ \sigma'_{\mathcal{AP}} &= b \cdot \sigma_{\mathcal{AP}} \\ P' &= b \cdot P \\ pk'_{\mathcal{AP}} &= b \cdot pk_{\mathcal{AP}} \\ C' &= b \cdot H(S) \cdot P \end{aligned} \tag{6.3}$$

4. $\mathcal{U}$ també calcula una ZKP no interactiva de Schnorr, que inclou la prova de possessió de la clau privada i l'acord sobre el màxim valor del comptador, escullin $r' \in_R \mathbb{F}_q^*$ i obtenint:

$$\begin{aligned} R' &= r' \cdot P \\ h' &= (max + 1) \cdot H(id_{\mathcal{U}}) \\ t' &= h' \cdot sk'_{\mathcal{U}} + r' \end{aligned} \tag{6.4}$$

5. $\mathcal{U}$ envia a $\mathcal{SP}$ la credencial anònima $\sigma'_{\mathcal{AP}}$, $pk'_{\mathcal{U}}$, $pk'_{\mathcal{AP}}$, P', C'. $\mathcal{U}$ també envia l'identificador únic $id_{\mathcal{U}}$, i la prova que posseeix la clau privada amb la ZKP no interactiva de Schnorr (R', h', t') per permetre la verificació de la credencial i l'identificador associat a ella.

6. $\mathcal{SP}$ ha de verificar que $pk'_{\mathcal{AP}}$ és realment $pk_{\mathcal{AP}}$ després de ser emmascarada. Per fer això, $\mathcal{SP}$ ha de comprovar que:

$$\begin{aligned} e(pk'_{\mathcal{AP}}, P) &= e(b \cdot pk_{\mathcal{AP}}, P) \\ &= e(pk_{\mathcal{AP}}, P)^b \\ &= e(pk_{\mathcal{AP}}, b \cdot P) \\ &= e(pk_{\mathcal{AP}}, P') \end{aligned}$$

7. si $pk'_{\mathcal{AP}}$ és correcte, $\mathcal{SP}$ pot verificar el següent:

$$\begin{aligned} e(C' + pk'_{\mathcal{AP}}, \sigma'_{\mathcal{AP}}) &= e(b \cdot H(S) \cdot P + b \cdot pk_{\mathcal{AP}}, b \cdot \sigma_{\mathcal{AP}}) \\ &= e(b \cdot (H(S) \cdot P + sk_{\mathcal{AP}} \cdot P), b \cdot (H(S) + sk_{\mathcal{AP}})^{-1} \cdot pk_{\mathcal{U}}) \\ &= e(b \cdot P \cdot (H(S) + sk_{\mathcal{AP}}), b \cdot (H(S) + sk_{\mathcal{AP}})^{-1} \cdot pk_{\mathcal{U}}) \\ &= e(P' \cdot (H(S) + sk_{\mathcal{AP}}), b \cdot (H(S) + sk_{\mathcal{AP}})^{-1} \cdot pk_{\mathcal{U}}) \\ &= e(P' \cdot (H(S) + sk_{\mathcal{AP}}), b \cdot (H(S) + sk_{\mathcal{AP}})^{-1} \cdot sk_{\mathcal{U}} \cdot P) \\ &= e(P', b \cdot sk_{\mathcal{U}} \cdot P)^{(H(S)+sk_{\mathcal{AP}}) \cdot (H(S)+sk_{\mathcal{AP}})^{-1}} \\ &= e(P', pk'_{\mathcal{U}}) \end{aligned}$$

8. $\mathcal{SP}$ també ha de verificar l'identificador únic $id_{\mathcal{U}}$ seguint els passos següents:

$$\begin{aligned} e(C' + pk'_{\mathcal{U}}, H(S) \cdot id_{\mathcal{U}}) &= e(b \cdot H(S) \cdot P + b \cdot pk_{\mathcal{U}}, H(S) \cdot H(S)^{-1} \cdot (sk_{\mathcal{U}} + H(S))^{-1} \cdot pk_{\mathcal{U}}) \\ &= e(b \cdot (H(S) \cdot P + sk_{\mathcal{U}} \cdot P), (H(S) + sk_{\mathcal{U}})^{-1} \cdot pk_{\mathcal{U}}) \\ &= e(P \cdot (H(S) + sk_{\mathcal{U}}), (H(S) + sk_{\mathcal{U}})^{-1} \cdot pk_{\mathcal{U}})^{b} \\ &= e(P, b \cdot pk_{\mathcal{U}})^{(H(S)+sk_{\mathcal{U}}) \cdot (H(S)+sk_{\mathcal{U}})^{-1}} \\ &= e(P, pk'_{\mathcal{U}}) \end{aligned}$$

9. $\mathcal{SP}$ ha de verificar també que $\mathcal{U}$ posseeix la clau privada i està d'acord amb el màxim d'usos:

$$\begin{aligned} t' \cdot P &= (h' \cdot sk'_{\mathcal{U}} + r') \cdot P \\ &= (max + 1) \cdot H(id_{\mathcal{U}}) \cdot sk'_{\mathcal{U}} \cdot P + r' \cdot P \\ &= r' \cdot P + (max + 1) \cdot H(id_{\mathcal{U}}) \cdot pk'_{\mathcal{U}} \\ &= R' + (max + 1) \cdot H(id_{\mathcal{U}}) \cdot pk'_{\mathcal{U}} \end{aligned} \tag{6.5}$$

10. Finalment, si totes les validacions són correctes, l'SC emmagatzema un objecte indexat amb $H(id_{\mathcal{U}})$ on enregistra el valor màxim del comptador, el valor del comptador (inicialitzat a 0) i la clau pública ofuscada de l'usuari $pk'_{\mathcal{U}}$.

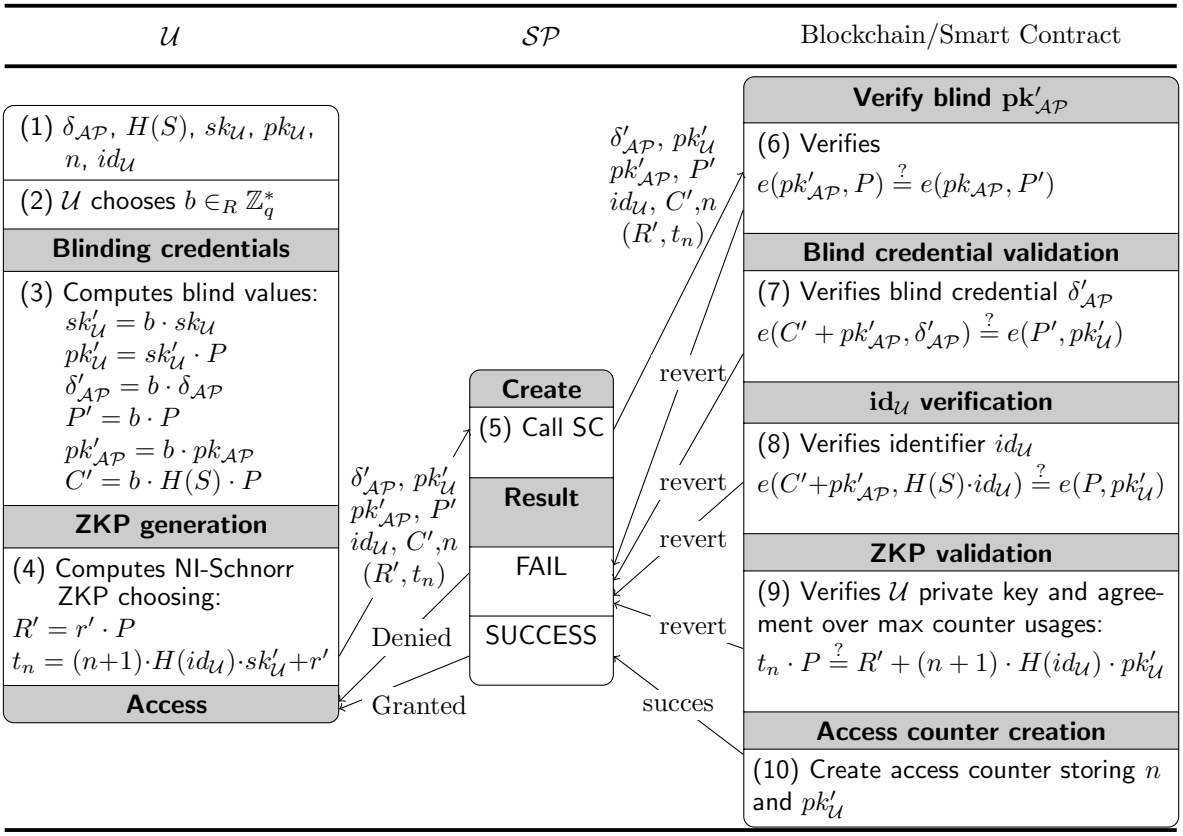

Figura 6.3: Intercanvi de missatges entre $\mathcal{U}$ i $\mathcal{SP}$ durant la fase de creació del comptador d'usos de la credencial.

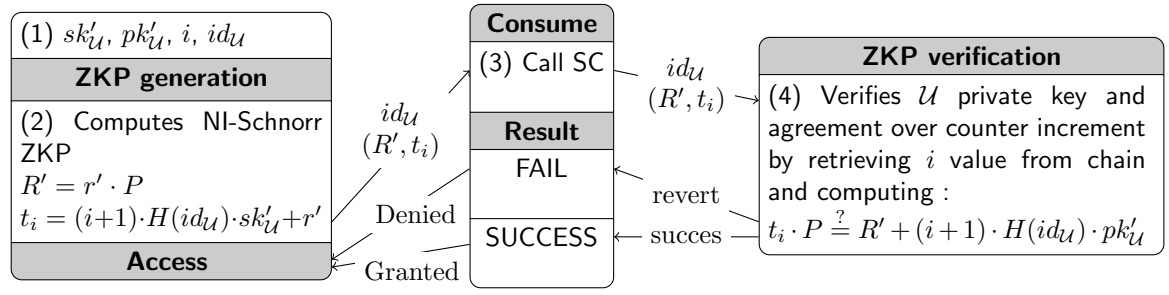

Figura 6.4: Visió gràfica del consum de les credencials.

Un cop creat el comptador d'ús ja es pot utilitzar el comptador d'usos de les credencials.

6.1.5 Consum de les credencials, increment del comptador ($\mathcal{U} \Longrightarrow \mathcal{SP}$)

A l'hora de consumir les credencials cal que, a més del seu identificador, l'usuari proporcioni la prova de coneixement nul que engloba el coneixement de la clau privada $sk'_{\mathcal{U}}$ associada i del següent valor del comptador. Aquesta es calcula

$$\begin{aligned} R' &= r' \cdot P \\ h_i &= (i+1) \cdot H(id_{\mathcal{U}}) \\ t_i &= h_i \cdot sk'_{\mathcal{U}} + r' \end{aligned} \tag{6.6}$$

$\mathcal{SP}$ fa una crida al mètode Consume del SC que realitza les següents comprovacions:

- A la iteració i comprova:

$$\begin{aligned} t_i \cdot P &= (h_i \cdot sk'_{\mathcal{U}} + r') \cdot P \\ &= (i+1) \cdot H(id_{\mathcal{U}}) \cdot sk'_{\mathcal{U}} \cdot P + r' \cdot P \\ &= r' \cdot P + (i+1) \cdot H(id_{\mathcal{U}}) \cdot pk'_{\mathcal{U}} \\ &= R' + (i+1) \cdot H(id_{\mathcal{U}}) \cdot pk'_{\mathcal{U}} \end{aligned} \tag{6.7}$$

  Aquesta comprovació serveix per comprovar tant que coneix la clau privada $sk'_{\mathcal{U}}$ associada a la clau pública $pk'_{\mathcal{U}}$ com que està d'acord amb el valor del comptador.

- Comprova que el valor següent sigui inferior o igual al límit establert en la creació el comptador.

Si tot és correcte i la transacció acaba bé, $\mathcal{SP}$ proveeix el servei i el comptador augmenta, si no, la transacció fa revert i l'$\mathcal{SP}$ denega el servei.

6.1.6 Cas d'ús detallat d'utilització del sistema

Es reprèn el cas d'ús enunciat a l'Apartat 4.2 de suggeriments per veure el procés amb més detall: L'usuari $\mathcal{U}$ que té ingressos inferiors al salari mínim té dret a 30 viatges en metro cada mes. En aquest cas l'$\mathcal{AP}$ pot ser l'agència tributària o el banc, i l'$\mathcal{SP}$ seria la gestora del metro. Es fixa l'àmbit com la cadena construïda amb la primera lletra del servei, el número del mes i l'any en què té dret a aquest servei, així "M0124"és l'àmbit per gener del 2024, "M0224"per febrer del 2024 i així en endavant. Els passos per un mes donat són:

1. $\mathcal{U}$ genera les seves claus $sk_{\mathcal{U}}$ i la corresponen $pk_{\mathcal{U}}$ que li poden servir sempre, o que pot canviar cada mes.

2. Al gener, calcula amb la Fórmula 6.2 el seu identificador únic per l'àmbit "M0124".

3. $\mathcal{U}$ va a l'$\mathcal{AP}$ s'identifica, prova que el mes anterior ha tingut ingressos inferiors al salari mínim i obté σ_{M0124} que li dona dret a 30 viatges de metro pel mes de gener. $\mathcal{AP}$ calcula σ_{M0124} amb la Fórmula 6.1, que lliga l'atribut a $sk_{\mathcal{U}}$.

4. L'atribut σ_{M0124} és conegut per l'$\mathcal{AP}$ que pot associar-lo a un usuari per controlar si li ha donat ja o no.

5. Per tant, abans de la seva utilització $\mathcal{U}$ ofusca σ_{M0124} i realitza els càlculs de la Formula 6.3 per obtenir les seves credencials anònimes de l'àmbit "M0124"que li dona dret a 30 viatges en metro durant el mes de gener del 2024.

6. Per començar a utilitzar els viatges, cal que obtingui un comptador. Per poder fer-ho a de presentar a l'$\mathcal{SP}$ les seves credencials anònimes calculades en el pas anterior, el seu $id_{\mathcal{U}}$ i una prova de que posseïx la clau privada. La prova t_{max+1} es calcula amb la Fórmula 6.4 amb i igual al número de viatges més un, és a dir t_{31}. Aquest prova es pot comprovar amb la Fórmula 6.7 tenint les credencials anònimes i l'$id_{\mathcal{U}}$. Aquestes comprovacions estan implementades en el mètode Create() del SC.

7. Amb això $\mathcal{SP}$, crida al mètode Create() de l'SC per crear un comptador vàlid per 30 accessos, ni un menys ni un més.

8. Només $\mathcal{U}$ pot utilitzar aquest comptador, utilitzant cada cop una t_i on i és el número d'utilitzacions emmagatzemat en el SC, junt al seu $id_{\mathcal{U}}$ i la seva clau pública. Si $\mathcal{U}$ manté aquestes proves t_i en secret, només ell pot consumir a cada cop el comptador. Això evita que qualsevol que no conegui la clau privada pugui fer avançar el comptador.

9. Un cop creat el comptador, per utilitzar-lo, $\mathcal{U}$ ha de calcular la t_i que correspon al valor actual del comptador, i enviar-la al $\mathcal{SP}$ junt al seu $id_{\mathcal{U}}$ i la seva clau pública ofuscada. Les comprovacions d'aquests valors estan implementades en el mètode Consume del SC i també que el comptador sigui inferior o igual al número màxim.

10. L'$\mathcal{SP}$ quan li arribi una petició de consum d'un comptador només cal que cridi a mètode Consume() del SC amb els paràmetres de la petició, si acaba bé pot donar accés, si es reverteix, quelcom ha fallat:

- no és la t_i que correspon
- intent de frau
- o el nombre d'usos ha arribat al limit

i ha de denegar l'accés.

6.2 Anàlisis de la seguretat

En aquesta secció es proporciona una anàlisi de seguretat del protocol proposat. En primer lloc, s'analitza la robustesa davant d'un usuari maliciós que intenta utilitzar credencials falses. També es verifiquen les propietats anònimes de l'esquema proposat. S'han analitzat diverses amenaces. Per realitzar l'anàlisi s'han fet certes assumpcions sobre $\mathcal{U}$:

- $\mathcal{U}$ pot forjar credencials falses.
- $\mathcal{U}$ pot ofuscar la credencial moltes vegades amb resultats diferents per utilitzar la credencial més d'una vegada.

En tots dos casos, es demostrarà que aquest comportament maliciós es detecta. També s'han fet assumpcions sobre les habilitats tant de $\mathcal{AP}$ com de $\mathcal{SP}$:

- $\mathcal{AP}$ coneix la identitat real de $\mathcal{U}$.
- $\mathcal{AP}$ i $\mathcal{SP}$ cooperen i recullen tots els missatges intercanviats amb $\mathcal{U}$.

Es demostrarà que, en qualsevol cas, $\mathcal{SP}$ no pot obtenir la identitat real de $\mathcal{U}$, i $\mathcal{AP}$ no pot saber quan $\mathcal{U}$ utilitza la seva credencial.

Infalsificació

En primer lloc, un usuari maliciós $\mathcal{U}$ no pot generar credencials ofuscades falses que no estiguin signades per $\mathcal{AP}$. Aquesta propietat està assegurada per l'ús de la signatura curta ZSS que té la propietat d'infalsificació. Més precisament, en cas que $\mathcal{U}$ generi una credencial falsa, aquesta no estaria signada correctament amb la clau $sk_{\mathcal{AP}}$, i la signatura falsa ZSS no passarà la validació realitzada per $\mathcal{SP}$ a la fase de presentació de credencials, la verificació

$$e(C' + pk'_{\mathcal{AP}}, \sigma'_{\mathcal{AP}}) = e(P', pk'_{\mathcal{U}})$$

fallaria. Si no fallés, significaria que la signatura curta ZSS no té la propietat d'infalsificabilitat, cosa que, com s'ha esmentat anteriorment, és falsa.

En segon lloc, un usuari maliciós $\mathcal{U}$ no pot generar un $id_{\mathcal{U}}$ que no estigui associat a un $\mathcal{AP}$ per a un àmbit determinat S, ja que aquest identificador no passaria la fase de validació realitzada per $\mathcal{SP}$. Més precisament, a la fase d'emissió de credencials, el càlcul de $id_{\mathcal{U}}$:

$$id_{\mathcal{U}} = H(S)^{-1}(sk_{\mathcal{U}} + H(S))^{-1} \cdot pk_{\mathcal{U}}$$

proporciona un identificador únic per a l'àmbit donat, S, i una clau privada, $sk_{\mathcal{U}}$. En cas que $\mathcal{U}$ generi un identificador diferent:

$$\tilde{id}_{\mathcal{U}} = H(S)^{-1} \cdot (\tilde{sk}_{\mathcal{U}}^{-1} + H(S))^{-1} \cdot \tilde{pk}_{\mathcal{U}}$$

$\mathcal{SP}$ ha de verificar C' i $\tilde{id}_{\mathcal{U}}$ amb el mateix $pk'_{\mathcal{U}}$; tanmateix, si $\mathcal{U}$ envia $pk'_{\mathcal{U}} = b \cdot pk_{\mathcal{U}}$ a la fase de presentació de credencials, la verificació realitzada per $\mathcal{SP}$ de $\tilde{id}_{\mathcal{U}}$ fallarà perquè

$$e(C' + pk'_{\mathcal{U}}, H(S) \cdot \tilde{id}_{\mathcal{U}}) \neq e(P, pk'_{\mathcal{U}})$$

D'altra banda, si $\mathcal{U}$ envia $\tilde{pk}'_{\mathcal{U}} = b \cdot \tilde{sk}_{\mathcal{U}} \cdot P$, $\mathcal{SP}$ pot detectar l'atac ja que

$$e(C' + pk'_{\mathcal{AP}}, \sigma'_{\mathcal{AP}}) \neq e(P, \tilde{pk}'_{\mathcal{U}})$$

Finalment, un usuari maliciós $\mathcal{U}$ no pot generar una clau pública falsa enofuscada $pk'_{\mathcal{AP}}$ associada a una $\mathcal{AP}$ falsa en l'àmbit $\mathcal{S}$, ja que no passarà la fase de validació realitzada per $\mathcal{SP}$. Un adversari $\tilde{A}$ que vol convèncer $\mathcal{SP}$ que una clau pública ofuscada falsa de $\mathcal{AP}$ és vàlida, ha de trobar X que $X \cdot \lambda \cdot pk'_{\tilde{\mathcal{AP}}} \equiv \lambda \cdot pk_{\mathcal{AP}}$, però això no és possible a causa del DLP (Problema 1 a la Secció 2.2.3), l'Inv-CDHP (Problema 4 a la Secció 2.2.3) i el DCDHP (Problema 3 a la Secció 2.2.3).

Anonimat de l'usuari

El protocol proposat protegeix la identitat de $\mathcal{U}$ de manera que un adversari $\tilde{A}$ no podrà obtenir-la encara que tingui accés a una sèrie de missatges i/o credencials generats per $\mathcal{AP}$ o processats per $\mathcal{SP}$ del mateix usuari. De fet, la identitat d'$\mathcal{U}$ està protegida encara que $\mathcal{AP}$ i $\mathcal{SP}$ cooperin.

En la fase d'emissió de credencials, $\mathcal{AP}$ no coneix el valor de $id_{\mathcal{U}}$ generat per $\mathcal{U}$ en l'últim pas del protocol ja que es calcula utilitzant la clau privada de $\mathcal{U}$.

A la fase de presentació de credencials, $\mathcal{U}$ utilitza un factor de ofuscat b per protegir la informació enviada a $\mathcal{SP}$, de manera que $\mathcal{SP}$ no pot obtenir la identitat de $\mathcal{U}$ a partir de la credencial ofuscada rebuda. L'ús de signatures ZSS curtes permet una verificació de seguretat adequada sobre dades ofuscades.

Pel que fa al valor $id_{\mathcal{U}}$, $\mathcal{SP}$ no el pot utilitzar per obtenir la identitat de $\mathcal{U}$, ja que, com s'ha comentat per al cas de $\mathcal{AP}$, $id_{\mathcal{U}}$ el genera $\mathcal{U}$ utilitzant la clau privada de $\mathcal{U}$.

Tingueu en compte, també, que la cooperació entre $\mathcal{AP}$ i $\mathcal{SP}$ no pot comprometre l'anonimat de $id_{\mathcal{U}}$. Com que $\mathcal{SP}$ només rep valors cecs, fins i tot si coopera amb $\mathcal{AP}$, no té possibilitats d'obtenir la identitat de l'usuari, a causa dels supòsits DLP, CDHP i DCDHP (Problemes 1, 2 i 3 respectivament a l'Apartat 2.2.3).

No enllaçabilitat de l'identificador entre àmbits

Donats n àmbits diferents, $S_1, S_2, \cdots, S_n$, un adversari que té coneixement dels identificadors corresponents per a un usuari determinat $\mathcal{U}$, denotats per $id_{\mathcal{U}}^{S_1}, id_{\mathcal{U}}^{S_2}, \cdots, id_{\mathcal{U}}^{S_n}$, no pot enllaçar

aquests identificadors ni obtenir la identitat de $id_{\mathcal{U}}$.

Aquesta propietat està assegurada per la definició d'identificador:

$$id_{\mathcal{U}}^{S_i} = H(S_i)^{-1}(sk_{\mathcal{U}} + H(S_i))^{-1} \cdot pk_{\mathcal{U}}$$

ja que obtenir $sk_{\mathcal{U}}$ o $pk_{\mathcal{U}}$, fins i tot amb el coneixement de $S_1, S_2, \cdots, S_n$, no és possible a causa del DLP (Problema 1 a l'Apartat 2.2.3).

Detecció de la reutilització de l'identificador dins del mateix àmbit

El protocol proposat és capaç de detectar la reutilització d'identificadors d'usuari dins del mateix àmbit. Aquesta propietat es basa en com es defineix $id_{\mathcal{U}}$.

Tenint en compte que tots els termes de l'expressió:

$$id_{\mathcal{U}} = H(S)^{-1} \cdot (sk_{\mathcal{U}}^{-1} + H(S))^{-1} \cdot pk_{\mathcal{U}}$$

es fixen per a un àmbit determinat S, de manera que l'$id_{\mathcal{U}}$ resultant serà el mateix per a un determinat $sk_{\mathcal{U}}$ i $pk_{\mathcal{U}}$. $\mathcal{SP}$ només necessita emmagatzemar $id_{\mathcal{U}}$ per detectar-ne la reutilització. A més, com que $id_{\mathcal{U}}$ no depèn del valor cec C', $id_{\mathcal{U}}$ serà el mateix per a diferents versions ofuscades de C.

Un usuari maliciós $\mathcal{U}$ no pot generar un $id_{\mathcal{U}}$ que no estigui associat a un $\tilde{sk}_{\mathcal{U}}$ i $\tilde{pk}_{\mathcal{U}}$, ja que aquest identificador no passarà la fase de validació realitzada per $\mathcal{SP}$. De la mateixa manera que a l'Apartat 6.2, $\mathcal{U}$ genera un identificador diferent:

$$\tilde{id}_{\mathcal{U}} = H(S)^{-1} \cdot (\tilde{sk}_{\mathcal{U}}^{-1} + H(S))^{-1} \cdot \tilde{pk}_{\mathcal{U}}$$

$\mathcal{SP}$ ha de verificar C' i $\tilde{id}_{\mathcal{U}}$ amb el mateix $pk'_{\mathcal{U}}$, però, si $\mathcal{U}$ envia $pk'_{\mathcal{U}} = b \cdot pk_{\mathcal{U}}$ a la fase de presentació de credencials, la verificació realitzada per $\mathcal{SP}$ de $\tilde{id}_{\mathcal{U}}$ fallarà, perquè

$$e(C' + pk'_{\mathcal{U}}, H(S) \cdot \tilde{id}_{\mathcal{U}}) \neq e(P, pk'_{\mathcal{U}})$$

D'altra banda, si $\mathcal{U}$ envia $\tilde{pk}'_{\mathcal{U}} = b \cdot \tilde{sk}_{\mathcal{U}} \cdot P$, $\mathcal{SP}$ pot detectar l'atac ja que

$$e(C' + pk'_{\mathcal{AP}}, \sigma'_{\mathcal{AP}}) \neq e(P, \tilde{pk}'_{\mathcal{U}})$$

Ús d'un comptador arbitrari sense creació

Les dades dels comptadors s'emmagatzemen a la cadena i només l'SC pot afegir aquestes dades després de passar amb èxit totes les comprovacions. Per utilitzar el comptador, s'ha de crear abans. Això garanteix que ningú pugui utilitzar un comptador arbitrari abans de la creació, on es realitzen comprovacions de credencials.

Un mal comportament de $\mathcal{SP}$ en la fase de creació del comptador

Per disseny, l'SC verifica t_n. Com que es calcula com:

$$t_n = (n+1) \cdot H(id_{\mathcal{U}}) \cdot sk_{\mathcal{U}} + r'$$

només pot ser proporcionat per algú en possessió de la clau privada, r' que aprova el valor de n.

Un atac de repetició

En associar cada t_i amb un valor de comptador i i incloure'l en el càlcul, es poden evitar els atacs de repetició perquè una prova vàlida t_i per a un valor de comptador de i esdevé invàlida per a un valor de comptador de $i+1$. Cada t_i s'invalida després de l'ús. Només els que tenen una clau privada poden calcular un t_{i+1} vàlid per al valor del comptador de $i+1$. Només l'SC pot alterar el valor d'un comptador després de passar amb èxit totes les comprovacions.

Un mal comportament de $\mathcal{SP}$

Aplicat per l'SC, sense conèixer la clau privada, ningú pot falsificar un t_i i que sigui vàlid. Per tant, un $\mathcal{SP}$ maliciós no pot incrementar el comptador sense l'acord de l'usuari que proporciona el t_i corresponent.

6.3 Avaluació dels protocols

El primer que s'ha avaluat ha estat les credencials, és a dir les modificacions a les credencials anònimes de l'Apartat 5.1 amb la introducció de l'àmbit i la validesa de l'$id_{\mathcal{U}}$. S'ha utilitzat com a base una implementació en Golang[1] que es podia trobar a GitHub[2], malauradament, ja no està disponible en línia. El llenguatge Golang ofereix suport per enters grans, corbes el·líptiques i aparellaments bilineals, base del desenvolupament del protocol. Aquesta implementació utilitza els quatre rols originals que són l'$\mathcal{U}$, l'$\mathcal{IV}$, el $\mathcal{CP}$ i el $\mathcal{SP}$, i ha desenvolupat un programa servidor que serveix una API REST que ofereix un servei per cada càlcul a realitzar. També disposa de dos programes més, un escrit en Java [57] que orquestra el protocol fent les diferents crides a API REST, i un darrer per generar el material criptogràfic necessari per dur a terme el protocol.

Per aquesta recerca s'ha reaprofitat, modificat o afegit serveis al servidor per cobrir tots els càlculs dels Apartats 5.1 a 5.3. S'han modificat també els rols que passen de quatre a tres com s'ha detallat a l'Apartat 5.1, que són l'$\mathcal{U}$, l'$\mathcal{AP}$ i el $\mathcal{SP}$.

Cada servei té un endpoint diferent i s'han separat els diferents serveis per rols del protocol. Les transferències de dades es duen a terme amb format JSON, tant paràmetres d'entrada com resultats. Els nombres es representen amb la cadena de caràcters de la seva representació en hexadecimal. Quan hi ha paràmetres d'entrada, la petició es fa tipus POST i els paràmetres són una estructura JSON dins del body del protocol; si no hi ha paràmetres, la petició és un

[1] URL: `https://go.dev/`.
[2] URL: `https://github.com/odib/privacy-preserving-credential-scheme-over-blockchain.git`.

GET amb el body buit. El resultat es retorna en el body en format JSON, i sempre són de tipus string. Les especificacions d'aquesta API REST són les següents:

6.3.1 Generació de claus

Aquesta funció l'invoca l'usuari per generar el seu parell de claus asimètriques.

Endpoint: /user/generateKey
Mètode: GET
Paràmetres: cap
Retorn: `{"priv":"string", "pub":"string"}`

No té cap paràmetre d'entrada i retorna una estructura JSON amb dos camps:

- *priv:* és una cadena de caràcters amb la representació de la clau privada codificada en hexadecimal.
- *pub:* és una cadena de caràcters amb la representació de la clau pública codificada en hexadecimal.

6.3.2 Generació de claus aparellades

Aquesta funció l'invoca l'usuari per generar les claus de l'aparellament biliniar $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$:

Endpoint: /user/generateKeyPairing
Mètode: GET
Paràmetres: cap
Retorn: `{ "priv":"string", "g1Pub":"string", "g2Pub":"string"}`

No té cap paràmetre d'entrada i retorna una estructura JSON amb dos camps:

- *priv:* és una cadena de caràcters amb la representació de la clau privada codificada en hexadecimal.
- *g1Pub:* és una cadena de caràcters amb la representació de la clau pública codificada en hexadecimal per al grup $\mathbb{G}_1$.
- *g2Pub:* és una cadena de caràcters amb la representació de la clau pública codificada en hexadecimal per al grup $\mathbb{G}_2$.

6.3.3 Generació d'identificador únic d'usuari

Aquesta funció l'invoca l'usuari per generar l'$id_{\mathcal{U}}$ a partir de l'àmbit, la seva clau privada i la seva clau pública del grup $\mathbb{G}_2$.

Endpoint: /user/generateId
Mètode: GET
Paràmetres: `{"scope":"string", "priv":"string", "g2Pub":"string"}`
Retorn: `{"idu":"string"}`

Com a paràmetres d'entrada necessita d'una estructura JSON amb els següents camps:

- *scope:* és una cadena de caràcters que identifica l'àmbit pel qual volem generar l'$id_{\mathcal{U}}$.
- *priv:* és una cadena de caràcters amb la representació de la clau privada codificada en hexadecimal.
- *g2Pub:* és una cadena de caràcters amb la representació de la clau pública codificada en hexadecimal per al grup $\mathbb{G}_2$.

I com a sortida retorna una estructura JSON que conté un únic camp $id_{\mathcal{U}}$ que és una cadena de caràcters amb la representació hexadecimal del seu valor.

6.3.4 Verificació de l'identificador únic d'usuari

Aquesta funció l'invoca qui vulgui verificar l'$id_{\mathcal{U}}$, ja sigui l'$\mathcal{AP}$ o el $\mathcal{SP}$, per verificar que l'$id_{\mathcal{U}}$ està associat a l'àmbit, al compromís, i a les dues claus públiques passades com a paràmetres. La particularitat d'aquesta funció és que també serveix si el compromís i les claus estan ofuscats amb el mateix factor. Per tant, es pot validar l'$id_{\mathcal{U}}$ associat a unes credencials i certificat ofuscat.

Endpoint: /ap/verifyIdu
Mètode: POST
Paràmetres:
```
{"scope":"string", "commit":"string, "idu":"string",
 "pubG1User":"string", "pubG2User":"string"}
```
Retorn:
```
{"verified":"bool"}
```

Com a paràmetres d'entrada necessita d'una estructura JSON amb els següents camps:

- *scope:* és una cadena de caràcters que identifica l'àmbit pel qual s'ha generat l'$id_{\mathcal{U}}$.
- *commit:* és una cadena de caràcters que amb la representació del compromís codificada en hexadecimal pel qual s'ha generat l'$id_{\mathcal{U}}$.
- *idu:* és una cadena de caràcters amb la representació de l'$id_{\mathcal{U}}$ codificat en hexadecimal.
- *pubG1User:* és una cadena de caràcters amb la representació de la clau pública codificada en hexadecimal per al grup $\mathbb{G}_1$.
- *pubG2User:* és una cadena de caràcters amb la representació de la clau pública codificada en hexadecimal per al grup $\mathbb{G}_2$.

Els paràmetres *commit*, *pubG1User* i *pubG2User* poden estar ofuscats però cal que el factor de d'ofuscació sigui el mateix per als tres. Com a sortida retorna una estructura JSON que conté un únic camp *verified* que és una cadena de caràcters amb la representació booleana del seu valor true/false.

6.3.5 Generació del certificat

Aquesta funció l'invoca l'$\mathcal{AP}$ per signar el compromís amb la seva clau privada i la pública a $\mathbb{G}_2$ de l'usuari per lligar-la a la seva identitat.

Endpoint: /ap/generateCertificate
Mètode: GET
Paràmetres: `{"commit":"string", "privAP":"string", "pubG2User":"string"}`
Retorn: `{"certificate":"string"}`

Com a paràmetres d'entrada necessita d'una estructura JSON amb els següents camps:

- *commit:* és una cadena de caràcters que amb la representació del compromís codificada en hexadecimal pel qual es vol generar el certificat.
- *privAP:* és una cadena de caràcters amb la representació de la clau privada d'$\mathcal{AP}$ codificada en hexadecimal.
- *pubG2User:* és una cadena de caràcters amb la representació de la clau pública codificada en hexadecimal per al grup $\mathbb{G}_2$.

I com a sortida retorna una estructura JSON que conté un únic camp *certificate* que és una cadena de caràcters amb la representació hexadecimal de la signatura.

6.3.6 Verificació del certificat

Aquesta funció l'invoca l'usuari per verificar que el certificat està associat al compromís, a la seva clau pública de $\mathbb{G}_2$ i signada per la privada de l'$\mathcal{AP}$ passats com a paràmetres.

Endpoint: /user/verifyCertificate
Mètode: POST
Paràmetres: `{"commit":"string", "certificate":"string", "pubG1AP":"string", "pubG2User":"string"}`
Retorn: `{"verified":"bool"}`

Com a paràmetres d'entrada necessita d'una estructura JSON amb els següents camps:

- *commit:* és una cadena de caràcters que amb la representació del compromís codificada en hexadecimal pel qual es vol verificar el certificat.
- *certificate:* és una cadena de caràcters amb la representació en hexadecimal de la suignatura que es vol verificar.
- *pubG1AP:* és una cadena de caràcters amb la representació del clau pública d'$\mathcal{AP}$ a $\mathbb{G}_1$ codificada en hexadecimal.
- *pubG2User:* és una cadena de caràcters amb la representació de la clau pública d'$\mathcal{U}$ codificada en hexadecimal per al grup $\mathbb{G}_2$.

Com a sortida retorna una estructura JSON que conté un únic camp *verified* que és una cadena de caràcters amb la representació booleana del seu valor true/false.

6.3.7 Ofuscació del certificat

Aquesta funció l'invoca l'usuari per ofuscar les seves credencials.

Endpoint: /user/blindCertificate

Mètode: POST

Paràmetres:
```
{"scope":"string", "certificate":"string", "pubG1AP":"string",
 "pubG1User":"string", "pubG2User":"string"}
```

Retorn:
```
{"blindC":"string", "blindCertificate":"string",
 "blindPubG1AP":"string", "blindPubG1User":"string",
 "blindPubG2User":"string", "blindGenerator":"string",
 "randomInv":"string"}
```

Com a paràmetres d'entrada necessita d'una estructura JSON amb els següents camps:

- *scope:* és una cadena de caràcters que identifica l'àmbit pel qual s'ha generat el certificat.
- *certificate:* és una cadena de caràcters amb la representació en hexadecimal de la signatura d'$\mathcal{AP}$.
- *pubG1AP:* és una cadena de caràcters amb la representació de la clau pública d'$\mathcal{AP}$ a $\mathbb{G}_1$ codificada en hexadecimal.
- *pubG1User:* és una cadena de caràcters amb la representació de la clau pública codificada en hexadecimal per al grup $\mathbb{G}_1$.
- *pubG2User:* és una cadena de caràcters amb la representació de la clau pública codificada en hexadecimal per al grup $\mathbb{G}_2$.

I retorna una estructura JSON amb els següents camps:

- *blindC:* és una cadena de caràcters amb la representació del compromís ofuscat i codificat en hexadecimal.
- *blindCertificate:* és una cadena de caràcters amb la representació de la signatura ofuscada i codificada en hexadecimal.
- *blindPubG1AP:* és una cadena de caràcters amb la representació de la clau pública d'$\mathcal{AP}$ ofuscada i codificada en hexadecimal per al grup $\mathbb{G}_1$.
- *blindPubG1User:* és una cadena de caràcters amb la representació de la clau pública d'$\mathcal{U}$ ofuscada i codificada en hexadecimal per al grup $\mathbb{G}_1$.
- *blindPubG2RUser:* és una cadena de caràcters amb la representació de la clau pública d'$\mathcal{U}$ ofuscada i codificada en hexadecimal per al grup $\mathbb{G}_2$.
- *blindGenerator:* és una cadena de caràcters amb la representació del generador ofuscat i codificat en hexadecimal del grup $\mathbb{G}_1$ codificat en hexadecimal.
- *randomInv:* és una cadena de caràcters amb la representació l'invers del factor de ofuscat.

6.3.8 Verificació del certificat ofuscat

Aquesta funció l'invoca l'$\mathcal{SP}$ per verificar les credencials anònimes.

Endpoint: /sp/verifyBlindCertificate
Mètode: POST
Paràmetres:
```
{"blindC":"string", "blindPubG1AP":"string", "blindPubG2User":"string",
 "blindCertificate":"string", "blindGenerator":"string"}
```
Retorn:
```
{"verified":"bool"}
```

Com a paràmetres d'entrada necessita d'una estructura JSON amb els següents camps:

- *blindC:* és una cadena de caràcters amb la representació de compromís ofuscat i codificat en hexadecimal.
- *blindPubG1AP:* és una cadena de caràcters amb la representació de la clau pública d'$\mathcal{U}$ ofuscada i codificada en hexadecimal per al grup $\mathbb{G}_1$.
- *blindPubG2User:* és una cadena de caràcters amb la representació de la clau pública d'$\mathcal{U}$ ofuscada i codificada en hexadecimal per al grup $\mathbb{G}_2$.
- *blindCertificate:* és una cadena de caràcters amb la representació de la clau pública ofuscada i codificada en hexadecimal per al grup $\mathbb{G}_1$.
- *blindGenerator:* és una cadena de caràcters amb la representació del generador ofuscat i codificat del grup $\mathbb{G}_1$ codificat en hexadecimal.

Com a sortida retorna una estructura JSON que conté un únic camp *verified* que és una cadena de caràcters amb la representació booleana del seu valor true/false.

6.3.9 Generació de conformitat

Aquesta funció l'invoca l'usuari per generar ZKP a partir del valor que es vol provar el coneixement i la seva clau pública a $\mathbb{G}_2$.

Endpoint: /user/generateZKP
Mètode: POST
Paràmetres:
```
{"value":"string", "pubG2":"string"}
```
Retorn:
```
{"A":"string", "t":"string", "pubSecret":"string"}
```

Com a paràmetres d'entrada necessita d'una estructura JSON amb els següents camps:

- *value:* és una cadena de caràcters amb la representació en hexadecimal del valor que es vol provar el coneixement.
- *pubG2:* és una cadena de caràcters amb la representació de la clau pública d'$\mathcal{U}$ codificada en hexadecimal per al grup $\mathbb{G}_2$.

I retorna una estructura JSON amb els següents camps:

- *A:* és una cadena de caràcters amb la representació d'un valor aleatori codificat en hexadecimal que serveix per calcular i verificar la ZKP.
- *t:* és una cadena de caràcters amb la representació d'un valor codificat en hexadecimal que és un valor públic que conté el secret utilitzat per verificar la ZKP.
- *pubSecret:* és una cadena de caràcters amb la representació d'un valor codificat en hexadecimal que és el valor del generador públic multiplicat pel secret és a dir és $secret * Generador$

Aquests tres valors representen la ZKP.

6.3.10 Verificació de conformitat

Aquesta funció la crida l'$\mathcal{SP}$ per verificar la ZKP a partir dels valors de la ZKP i la clau pública a $\mathbb{G}_2$ de l'$\mathcal{U}$.

Endpoint: /sp/verifyZKP
Mètode: POST
Paràmetres: `{"A":"string", "t":"string", "pubG2":"string", "pubSecret":"string"}`
Retorn: `{"verified":"bool"}`

Com a paràmetres d'entrada necessita d'una estructura JSON amb els següents camps:

- *A:* és una cadena de caràcters amb la representació d'un valor aleatori codificat en hexadecimal que serveix per calcular i verificar la ZKP.
- *t:* és una cadena de caràcters amb la representació d'un valor codificat en hexadecimal que és un valor públic que conté el secret utilitzat per verificar la ZKP.
- *pubG2:* és una cadena de caràcters amb la representació de la clau pública d'$\mathcal{U}$ codificada en hexadecimal per al grup $\mathbb{G}_2$.
- *pubSecret:* és una cadena de caràcters amb la representació d'un valor codificat en hexadecimal que és el valor del generador públic multiplicat pel un secret és a dir és $secret * Generador$

Com a sortida retorna una estructura JSON que conté un únic camp *verified* que és una cadena de caràcters amb la representació booleana del seu valor true/false.

6.4 Llibreries i programes per la validació

Tots aquests serveis s'han desenvolupat instrumentats de tal manera que registren el temps consumit a qualsevol de les operacions. Juntament amb el servidor ja esmentat en l'apartat anterior, s'ha desenvolupat una llibreria en llenguatge Python[3] per consumir aquests serveis, el

[3] URL: `https://www.python.org`.

seu codi es troba al Llistat 7.1. Amb aquesta llibreria s'han desenvolupat diversos programes per provar tant el "Happy Path" com conductes malintencionades.

En primer lloc, s'ha implementat el "Happy Path", el Llistat 7.2 és un exemple d'un programa que executa cada pas del protocol de manera seqüencial, a fi i efecte de validar tota la seqüència de manera correcta. A banda d'aquest "Happy Path" també s'ha volgut provar com es detecten els intents de mal ús.

Per exemple, un intent fraudulent d'ús en diferents àmbits, el Llistat 7.3 presenta un programa que tipifica un cas de frau on un usuari maliciós intenta utilitzar un $id_{\mathcal{U}}$ associat a un àmbit en un altre àmbit.

Una altra possible mala conducta és un intent fraudulent d'ús amb diferents claus. El Llistat 7.4 presenta un programa que tipifica un cas de frau on un usuari maliciós intenta utilitzar un $id_{\mathcal{U}}$ calculat amb una parella de claus i presentant-ne unes diferents.

6.5 Mesures sobre el protocol

Es vol verificar que el protocol dissenyat permet una autenticació anònima enllaçable dins d'un àmbit donat, amb un sobrecost raonable que permet la seva utilització en un entorn real. Per plantejar un escenari per validar aquesta implementació, es necessita:

- Un servidor web desenvolupat que implementa una Application Programming Interface (API) Representational State Transfer (REST) oferint els serveis criptogràfics necessaris en cada etapa del protocol utilitzant la llibreria desenvolupada en l'Apartat 6.4.
- Un client que realitza crides REST al servidor seqüenciant l'execució del protocol.

Els programes han estat desenvolupats en llenguatge Golang per diversos motius. Aquesta elecció va ser fonamentada en la necessitat de crear una base robusta que facilités la integració en fases posteriors del projecte, ja que la intenció original era utilitzar la tecnologia blockchain *Hyperledger Sawtooth*[4] per a la gestió de les credencials i la verificació de l'autenticació. No obstant això, com s'explicarà més endavant, aquesta línia de desenvolupament va ser abandonada a mesura que el projecte evolucionava.

El sistema es compon de dos programes principals:

1. **Programa servidor**: Aquest servidor s'encarrega d'implementar totes les operacions criptogràfiques que es realitzen en cada etapa del protocol. Aquestes operacions inclouen la creació i verificació de les credencials, la validació dels processos d'autenticació, així com la gestió de les interaccions entre els diferents components del sistema. Les operacions es publiquen mitjançant serveis REST, la qual cosa permet la seva fàcil accessibilitat i reutilització en altres parts del sistema. Les operacions implementades són les que es mostren a les Figures 6.2 i 6.3.
2. **Programa client**: El client interactua amb el servidor mitjançant crides REST. Aquest programa està dissenyat per seqüenciar una execució completa del protocol, des de la

[4] URL: `https://www.lfdecentralizedtrust.org/hyperledger-sawtooth-1-0`.

generació inicial de les claus criptogràfiques fins a l'accés a un servei desitjat. Això permet comprovar la correcta implementació de totes les etapes del protocol, assegurant que l'autenticació es realitza de manera segura i enllaçada a l'àmbit definit.

Per dur a terme les proves d'implementació, es va utilitzar el següent equipament:

- **Servidor**: Un Mac Mini amb un processador Intel i7 de 4 nuclis a 2GHz i 16GB de memòria RAM. Aquest servidor s'encarrega d'executar el servidor web que ofereix els serveis REST per al procés d'autenticació i gestió de les credencials.
- **Client**: El programa client es va executar en un MacBook Pro amb un processador Intel i7 de 4 nuclis a 2.7GHz i 8GB de memòria RAM. Aquest dispositiu s'encarrega d'interactuar amb el servidor mitjançant les crides REST per seqüenciar el protocol i simular l'intercanvi d'informació entre els components del sistema.

Tots dos dispositius estan connectats a través d'una xarxa Ethernet d'1 Gbit, assegurant així una connexió ràpida i estable per a les comunicacions entre el client i el servidor durant les proves, de manera que es minimitza qualsevol retard en la transmissió de dades que pogués afectar els resultats de la validació del protocol.

El programari client efectuarà les següents accions cridant al servei adequat en cada cas:

1. Fa una petició REST al servei de generació de claus i emmagatzemar la resposta.
2. Amb les claus obtingudes al pas anterior fa una petició REST al servei de generació de credencials (la comprovació de la identitat s'omet per raons òbvies , és un pas manual) i les emmagatzema.
3. Amb el servei REST de verificació de credencials, verifica les credencials obtingudes al pas anterior.
4. Cega les credencials emmagatzemades invocant el servei REST de generació de credencials ofuscades i les emmagatzema.
5. Fa una petició REST al servei de verificació de credencials ofuscades i comprova que les obtingudes al pas anterior són vàlides.
6. Amb la credencial emmagatzemada, genera l'identificador únic i el valida fent una petició REST al servei de validació d'identificador.
7. Fa una crida al servei de generació de ZKP per obtenir la ZKP i la valida amb el servei de verificació de ZKP.

Aquest seguit de crides les realitza 100 vegades, i el servidor emmagatzema les mètriques de cada una de les crides.

6.5.1 Resultats

A la Taula 6.1 es recull la mitjana dels temps mesurats per cada operació sobre les 100 mostres. Amb aquest resultat podem estimar el cost a nivell temporal de les dues fases del protocol com veiem a la Taula 6.2.

Etapa	Mitja temps
(a) key generation	$38ms$
(b) credencial generation	$11ms$
(c) credencial verification	$57ms$
(d) blind credencial generation	$37ms$
(e) verify blind $pk'_{\mathcal{AP}}$	$15ms$
(f) blind credencial validation	$55ms$
(g) identifier validation	62
(h) ZKP generation	$< 1ms$
(i) ZKP validation	$< 1ms$

Taula 6.1: Mitjanes dels temps de cada càlcul per cada operació per 100 mostres

Fase del protocol	Mitja de temps
Generació $(a + b + c)$	$109ms$
Presentació $(d + e + f + g + h + i)$	$171ms$

Taula 6.2: Mitjanes dels temps de cada càlcul per cada fase del protocol per 100 mostres

6.5.2 Conclusions

En conclusió, la solució proposada és viable i compleix els objectius establerts inicialment. El protocol d'autenticació anònima ha estat implementat amb èxit, demostrant la capacitat de preservar la privadesa de l'usuari tot garantint la verificació i enllaç de les credencials dins d'un àmbit específic. No obstant això, quan es va passar a la fase d'implementació sobre la tecnologia blockchain escollida inicialment, Hyperledger Sawtooth, vam detectar diverses limitacions que van afectar la seva idoneïtat per al projecte.

En primer lloc, Hyperledger Sawtooth no compleix plenament els requisits de transparència, descentralització i estandardització que s'havien establert al començament del projecte. Tot i que Sawtooth és una plataforma potent per a la creació de solucions amb tecnologia blockchain privades, les seves característiques no són totalment compatibles amb les necessitats d'un sistema que busca garantir la total transparència i descentralització dels processos de validació i autenticació, especialment en un entorn com el de la tecnologia blockchain pública.

A més, un altre desavantatge important de Sawtooth va ser la limitació que presenta la codi-

ficació de SC en el llenguatge de programació Golang. Aquesta restricció fa que el desplegament de SC es vegi restringit a un conjunt de tecnologies blockchains específiques que suporten aquest llenguatge, la qual cosa limita notablement la flexibilitat a l'hora de seleccionar la plataforma blockchain adequada.

Per aquest motiu, vam decidir optar per un llenguatge més estàndard i àmpliament adoptat en el món de la tecnologia blockchain: Solidity. Solidity és el llenguatge de programació més utilitzat per al desenvolupament de SC a la plataforma Ethereum, que és reconeguda com la referència principal per a aquest tipus de tecnologia blockchain pública, descentralitzada i flexible.

Finalment, Ethereum ha estat escollida com a tecnologia blockchain de desplegament, ja que compleix tots els requisits per al projecte, incloent-hi la descentralització, la transparència, i la capacitat d'executar contractes intel·ligents de manera segura i eficaç mitjançant Solidity. Així, Ethereum s'ha consolidat com la millor opció per implementar la solució de manera robusta, escalable i amb un gran suport dins de la comunitat de la tecnologia blockchain.

6.6 Validació dels protocols sobre Ethereum

Havent validat el protocol amb la implementació en Golang, s'havia de codificar en Solidity per poder desplegar-lo en una blockchain Ethereum com marquen els requisits no funcionals. El llenguatge Solidity manca de suport per a corbes el·líptiques i tot buscant possibles implementacions es va descobrir la EIP-196[5] que suggereix afegir contractes precompilats per a la suma i la multiplicació escalar en una corba el·líptica específica per a l'aparellament. Això, al seu torn, es pot combinar amb EIP-197[6] per verificar els zkSNARK [58] als contractes intel·ligents d'Ethereum.

6.6.1 Implementació del protocol de credencials anònimes i l'identificador únic d'usuari sobre Ethereum

Això ha permès poder implementar un conjunt de funcions auxiliars i tipus en Solidity, que utilitza crides a aquests contractes precompilats que es pot veure en el Llistat 7.5. Aquest llistat defineix:

- unes estructures de dades pels punts pertanyents al grup $\mathbb{G}_1$ i unes estructures de dades pels punts pertanyents al grup $\mathbb{G}_2$.
- els punts generadors de cada grup P_1 per $\mathbb{G}_1$ i P_2 per $\mathbb{G}_2$.
- un mètode per empaquetar i un per desempaquetar els punts de cada grup.
- l'operació de negatiu a $\mathbb{G}_1$ $(P + negatiu(P) = 0)$
- l'operació de suma a $\mathbb{G}_1$.

[5] URL: `https://eips.ethereum.org/EIPS/eip-196`.
[6] URL: `https://eips.ethereum.org/EIPS/eip-197`.

- l'operació de multiplicació a $\mathbb{G}_1$ per un escalar.
- l'operació de comprovació del pairing de la forma

$$e(P_1[0], P_2[0]) * ... * e(P_1[n], P_2[n]) = 1$$

- l'operació de comprovació de paring que ens interessa pel protocol que aconseguim fent

$$e(P_1[0], negatiu(P_1[0]) * e(P_2(), P_2()) = 1$$

Havent definit aquestes funcions i tipus es pot codificar un SC que validi les credencials anònimes de l'Apartat 5.1, l'$id_{\mathcal{U}}$ de l'Apartat 5.2 i per verificar que el certificat ofuscat de l'$\mathcal{AP}$ és el correcte, aquestes implementacions es troben al Llistat 7.6. Com l'EIP-197 només verifica que el producte del pairing sigui igual a 1, per poder comprovar la igualtat de les Fórmules 5.6, 5.1 i 5.3 cal calcular el producte amb l'invers a un dels termes.

6.6.2 Implementació del comptador d'usos de credencials anònimes sobre Ethereum

Per emmagatzemar el nombre d'usos a la cadena cal una estructura de dades com la del Llistat 6.1

```
struct tCounter {
    uint count;
    uint max;
}
mapping (bytes32 => uint) public id2index; // Address to Counter
mapping (uint => tCounter) public counters;
uint totalt;
```

Llistat 6.1: Estructure de magatzem de comptadors

A part de les validacions de credencial, cal implementar també la verificació de les proves de consentiment de la Fórmula 5.5; aquesta es troba en el Llistat 7.7 Finalment, s'han implementat els mètodes inherents als comptadors d'usos amb les comprovacions necessàries que ofereixen les funcions exposades en els apartats anteriors amb els *require* pertinents. En el Llistat 7.8 podem veure la implementació.

6.6.3 Client d'Ethereum

Per poder validar el protocol un cop desplegat en una blockchain Ethereum privada, s'ha desenvolupat un programa client. Aquest programa d'una banda serà l'eina que ens permetrà executar les operacions criptogràfiques com són la generació i/o verificació de claus, signatures i proves de consentiment. Una de les implementacions de referència d'Ethereum és geth[7] que es basa en la llibreria Go-Etherium[8]. Per tant, sembla bona idea utilitzar el llenguatge Golang per fer la implementació del client que farà les crides a l'SC implementat.

[7] URL: `https://geth.ethereum.org/`.
[8] URL: `https://github.com/ethereum/go-ethereum`.

6.6.4 El moneder

S'ha dissenyat una estructura de dades en format JSON que representa un moneder on el programa client emmagatzema i recupera el material criptogràfic en cada pas. L'estructura de dades es mostra en el Llistat 7.9, que es serialitza amb el JSON del Llistat 6.2.

Veiem que el moneder com a tal està dividit en diferents estructures de dades amb funcions ben específiques. Aquest tipus d'estructures són:

- **KeyPairing**: és una estructura que conté el material criptogràfic, o sigui el conjunt de clau privada i claus públiques associades, definits dins de l'aparellament bilineals i està format pels següents camps:
  - **priv**: és una cadena de caràcters amb el valor hexadecimal de la clau privada.
  - **g1pub**: és una cadena de caràcters amb el valor hexadecimal de la clau pública a $\mathbb{G}_1$.
  - **g2pub**: és una cadena de caràcters amb el valor hexadecimal de la clau pública a $\mathbb{G}_2$.
- $id_{\mathcal{U}}$ és una estructura que conté el material criptogràfic referent al $id_{\mathcal{U}}$ i està formada pels següents camps:
  - **id**: és una cadena de caràcters amb el valor hexadecimal de l'$id_{\mathcal{U}}$.
  - **idG1**: és una cadena de caràcters amb el valor hexadecimal del producte de l'$id_{\mathcal{U}}$ pel generador de $\mathbb{G}_1$.
  - **idG2**: és una cadena de caràcters amb el valor hexadecimal del producte de l'$id_{\mathcal{U}}$ pel generador de $\mathbb{G}_2$.
  - **C**: és una cadena de caràcters amb el valor hexadecimal del compromís associat a l'$id_{\mathcal{U}}$.
- **ABCred**: és una estructura que conté el material criptogràfic que defineix una credencial anònima en aquest protocol i està formada pels següents camps:
  - **bC**: és una cadena de caràcters amb el valor hexadecimal del compromís ofuscat.
  - **bcert**: és una cadena de caràcters amb el valor hexadecimal del certificat ofuscat emès per $\mathcal{AP}$.
  - **bg1Ap**: és una cadena de caràcters amb el valor hexadecimal de la clau pública ofuscada de l'$\mathcal{AP}$.
  - **bg1User**: és una cadena de caràcters amb el valor hexadecimal de la clau pública ofuscada l'usuari a $\mathbb{G}_1$.
  - **bg2User**: és una cadena de caràcters amb el valor hexadecimal de la clau pública ofuscada l'usuari a $\mathbb{G}_2$.
  - **bG1**: és una cadena de caràcters amb el valor ofuscat en hexadecimal del generador de $\mathbb{G}_1$.

  - **bG2**: és una cadena de caràcters amb el valor ofuscat hexadecimal del generador de $\mathbb{G}_2$.

- **Counter**: és una estructura que conté el material criptogràfic necessari per poder utilitzar els comptadors i està formada pels següents camps:

  - **idc**: és una cadena de caràcters amb el valor hexadecimal de l'identificador del comptador retornat a l'hora de la creació.
  - **zkp**: és una llista ordenada de les proves a utilitzar a cada iteració com a proves de consentiment.

Un cop vistos els tipus de cada estructura, veiem el contingut dels camps:

- **Scope**: és una cadena de caràcters que identifica per quin àmbit s'ha creat el material criptogràfic que conté el moneder.
- **G1AP**: és una cadena de caràcters amb el valor hexadecimal de la clau pública a $\mathbb{G}_1$ de l'$\mathcal{AP}$.
- **keys**: és un camp de tipus KeyPairing que conté les claus generades a la fase inicial del protocol.
- **Idu**: és un camp de tipus Idu que conté el material criptogràfic per a l'àmbit definit en el camp **Scope** necessari per utilitzar l'$id_{\mathcal{U}}$.
- **Cred**: és un camp de tipus Cred que conté les credencials anònimes per a l'àmbit definit en el camp **Scope**.
- **Counter**: un cop creat un comptador conté el seu identificador i les proves de consentiment associades.


```
{
 "scope": "s",
 "g1AP" : "047df9eec1cc96950b5c0f2a80d801da72bd3ba8252a135b6970c3aecc019e1503f7f72fb
           e6ee5fa5658e75e1931649bc84049109ca2be65a52a2f3c59a58ace",
 "keys" : {
            "priv": "1c79e5afdbfdcf3af9cb745610aef750f70735c608945f4ad17e1bf753c85f9c",
            "g1pub": "22cba1075088c424c3199c18b990bfd7c8fda3e8c018f56097f6b9f615f7fe55
                      198318cea561e3b2492365b747c822214c0221771e71fdf399983f42522e96be",
            "g2pub": "0af3704285b530514a71b96938ef9bd1529cbad1b85d5f4d672183ec89ef42ed16
                      bd84734acc074b9eb8e2daf6bf6287aa838d5d0680365cc65be0528987a73a1c3a
                      96e737db846d8728df41178898a9b3171bf149d0b5aa719edc120734aed62148c9
                      02b9e77beb3778e9f6dca032dce97c7ae80cfc6aa483927335add57349"
          },
 "idu"  : {
          "id"  : "14ce999856e31b8b91339daf81fc8e659b03928f1430e24813d45583d46ccc5",
          "idG1" : "2ab96d02dd484a84e426f9e8aa49b9a8c6b1a52f0be7e2141f22a4bc5f3dbe4d1be8
                    bd7e9847fb90322be5609aea23c22d51f387dac801ab7c7ea2703380544",
          "idG2" : "2dab14360d786a595300a5356339ba4fed3574a0a32b7998ac1b2c7bc0501ccf1f9a
                    146269d93e8e617365fb6aa577517f2ee4ae759a0c034592315e1b4d9245305f05c9
                    e8eb12e815cd47494faff21f2b80bd25e779f7d6ce8a2134ed5e76b72d607dd371be
                    9a0914e8ebe79b2f074bbdbebad061d1955c6bd9fc4f11c78558",
            "C"  : "098d172fa1c40f74d27b99c2e6c11920590a42446d23f205d5f602f87b98186b158c
```

```
                    d81da1cad26d22b795ccfb85a0c5116c0f349a0a26d67b132f6ade57dc0e"
        },
"cred" : {
          "bC"    : "015dc155fd0febd03fa91dfbe5a0b650fe504598a1746b883bd0925a15762a6b0
                     e1480540ea29d0b4d5041c16ddb4488c9b3acb94b4d0a3728d79492b1744e0a",
          "bCert" : "0c0d2aebcf7c281571bde3a55c84170e4c96a2c5d8878a1cd9628701861ffada1
                     35850aeb78c814d6b00ab5b3a755a182c40bd82835f77610cf16a3b1a3688f825
                     4f1c12ac020bc69d50f850ae05663b42ee106ab72fb52a26c80aa0622d27805a1
                     ec496f2683a155da3ba90cfdf3465037b30703f0b714a8e0f5b5a2e51b5c",
          "bg1AP" : "18b8aed5e539b9fd16c2ab0c3ce0133da370016fc3b7329132c8c22c9eda8bb31
                     48867d0a76c6dfd1707170c671d47dbd3838a141d79e538bb84aba513cb4ca0",
          "bpriv" : "326f5a9fdd9cc076cc37c51a3bb490b271859dbd66f78fe0f315551ac991e251d
                     26813b3ea32c5b483a5a1db7d7bafc8667dd193605ceeb9991bb9f0894ccac",
          "bg1User":"0ef0423e83552f21b4633a64162a40964f58f853d93008759887f6fd6a4869cd1
                     baf58099729e5b4fd761628d7ef750ffe73bd79e3db7e1132efc73157a855fc",
          "bg2User":"1b611cfb5fc546cb575ba8a08cceef536f79feeab6f7bedf7ebb76df5eca73e02
                     b24a111a041c32d416ac84899f8d3df8b922995083e6086583eddaea6e5352c04
                     4e311740363fc197aaa88234468bcdedc394337de9647f5ac30ae4bb9b3d51101
                     3b70aec5777c67afd2191afde5db1ac660cf6be77129ecd0a36c789111b66",
          "bG1"   :"07ddd01ae05c112dd998563d4341ce056804b4831f36d4d166fc829f3a8e82d6
                    0d2054ebeb2009592828e57786c555deb486e73e5928d9f9b1668329d5853a98",
          "bG2"    :"15e762cf2c822f0c938985297a4e704aeaea6be0b88120912997ad87e08bc0561
                     0e0e6592120327117e62f264d67597fc12140dd269e8a712363fa7d1346369506
                     da6f38f3939c24fffc3a81f3e58322a5e7f5ecb9ff54abe3e1773c73b3aa64024
                     a7737c8b27ada1aace442942d57a8dcfdcd71997021a86f5b2b8414ee04d3"
},
"counter": {
  "idc": "063c1ad38971dc994455b74b1c8f29c625308162690f13d7f50841339ca00af72cc7
          d597e50bbf49f78ce398a5e9da3d86e1b88edd08b557a94f6a2124f2ad41",
  "zkp":[2187423178744963005239343355865496178076698086589854898801484291950513733569 4,
         6017236228401575575419689809518743154463178614353340766498220323377252815799,
         1204848354119279632069235180563979961670774076322416688867980191382517679152 1,
         1807973085398401706596501380176085607895230291209499301086138350427310076724 3,
         2222735294935962588991270052624637452648500660549784789344760908145216247348,
         8253982607727183334263932048745693914893062809420610911526342498593140223070]
}
}
```


Llistat 6.2: Estructura de dades del wallet en JSON

6.6.5 El programa client

En l'Apartat 6.3 ja s'han desenvolupat totes les funcions necessàries per implementar les eines de generació del material criptogràfic pel protocol, per tant s'han reaprofitat i s'han implementat la connexió a Ethereum i la crida als SCs. És un programa de línia de comandes amb múltiples nivells; la forma de cridar-lo és:

```
$ ./client <cmd_l1> <cmd_l2> [ <cmd_l3> ] <params cmd> [-w <nom fitxer>]
```

Cada opció de nivell 1 es desglossa amb n de nivell dos i cada opció de nivell 2 es desglossa, si s'escau, al seu torn en m opcions de nivell 3. Al primer nivell hi han dues opcions: **crypto** i **counter**.

L'opció **crypto** aglutina totes les opcions per generar i verificar material criptogràfic, té les següents dues opcions amb els seus subnivells: **generate** i **verify**.

La combinació **crypto generate** s'encarrega de la generació i té els següents paràmetres:

- **all**: genera tot el material, necessita com a paràmetres en l'ordre que apareixen:
  - una cadena de caràcters que es l'àmbit pel que es vol generar el material criptogràfic.
  - una cadena de caràcters en hexadecimal amb la clau privada d'$\mathcal{AP}$.
  - una cadena de caràcters en hexadecimal amb la clau pública d'$\mathcal{AP}$ en el grup $\mathbb{G}_1$.
  - un número que és el nombre d'usos que es desitja ja que generarà també les proves de consentiment.
- **cert**: signa el certificat amb la clau d'$\mathcal{AP}$, necessita com a paràmetres en l'ordre que apareixen:
  - una cadena de caràcters en hexadecimal amb el compromís de les credencials.
  - una cadena de caràcters en hexadecimal amb la clau privada d'$\mathcal{AP}$.
  - una cadena de caràcters en hexadecimal amb la clau pública de l'usuari propietari de les credencial en el grup $\mathbb{G}_2$.
- **idu**: genera l'$id_{\mathcal{U}}$, necessita com a paràmetres en l'ordre que apareixen:
  - una cadena de caràcters que es l'àmbit pel que es vol generar el material criptogràfic.
  - una cadena de caràcters en hexadecimal amb la clau privada de l'usuari.
  - una cadena de caràcters en hexadecimal amb la clau pública de l'usuari en el grup $\mathbb{G}_1$.
  - una cadena de caràcters en hexadecimal amb la clau pública de l'usuari en el grup $\mathbb{G}_2$.
- **kp**: genera un conjunt de clau de l'aparellament bilineal, no necessita paràmetres.

Totes les funcions de generació mostren per pantalla el material generat, i si s'especifica l'opció **-w**, a més, guarda aquest material en el moneder amb el nom indicat en el paràmetre.

La combinació **crypto verify** s'encarrega de la verificació i té els següents paràmetres:

- **cert**: verifica el certificat signat per l'$\mathcal{AP}$, necessita com a paràmetres en l'ordre que apareixen:
  - una cadena de caràcters que es l'àmbit pel que es vol generar el material criptogràfic.
  - una cadena de caràcters en hexadecimal amb el certificat emès per $\mathcal{AP}$.
  - una cadena de caràcters en hexadecimal amb la clau pública de l'usuari en el grup $\mathbb{G}_1$.
  - una cadena de caràcters en hexadecimal amb la clau pública de l'usuari en el grup $\mathbb{G}_2$.
- **idu**: verifica l'$id_{\mathcal{U}}$, necessita com a paràmetres en l'ordre que apareixen:
  - una cadena de caràcters que es l'àmbit pel que es vol generar el material criptogràfic.
  - una cadena de caràcters en hexadecimal amb compromís de les credencials.

  - una cadena de caràcters en hexadecimal amb l'$id_{\mathcal{U}}$.
  - una cadena de caràcters en hexadecimal amb la clau pública de l'usuari en el grup $\mathbb{G}_1$.
  - una cadena de caràcters en hexadecimal amb la clau pública de l'usuari en el grup $\mathbb{G}_2$.

Cal notar que degut a les propietats dels aparellaments bilineals sobre corbes el·líptiques en què es basa el protocol, la verificació es pot fer tant amb tots els valors originals com amb tots els valors ofuscats, excepte l'àmbit i l'$id_{\mathcal{U}}$, és clar.

L'opció **counter** desplega i interactua amb l'SC desplegat a Ethereum. Per totes les operacions s'utilitza el mateix compte ethereum guardat en un fitxer. A partir dels contractes compilats es genera el codi Golang amb la utilitat **abigen** que forma part de la suite de Go-Ethereum[9]. A més, el paràmetre **-w** és obligatori per totes les operacions excepte **deploy** ja que necessita un moneder per obtenir el material criptogràfic necessari per interactuar amb l'SC.

- **deploy**: desplega l'SC a la blockchain privada Ethereum, no necessita cap paràmetre. Guarda l'adreça del contracte en un fitxer per després poder utilitzar-lo.
- **create**: crea un comptador d'usos, necessita com a paràmetres en l'ordre que apareixen:
  - una cadena de caràcters en hexadecimal amb l'adreça de l'SC a la cadena.
  - una cadena de caràcters en hexadecimal amb la clau privada del compte Ethereum.
  - el màxim número d'usos.
- **consume**: utilitza un nou ús de les credencials anònimes, si fa revert l'$\mathcal{SP}$ ha de negar el servei, necessita com a paràmetres en l'ordre que apareixen:
  - una cadena de caràcters en hexadecimal amb l'adreça de l'SC a la cadena.
  - una cadena de caràcters en hexadecimal amb la clau privada del compte Ethereum.
- **get**: retorna el valor del comptador emmagatzemat a la cadena, necessita com a paràmetres en l'ordre que apareixen:
  - una cadena de caràcters en hexadecimal amb l'adreça de l'SC a la cadena.
  - una cadena de caràcters en hexadecimal amb la clau privada del compte Ethereum.

6.7 Mesures sobre Ethereum

El primer que es vol verificar, és que és possible fer la implementació del protocol dissenyat i dels comptadors amb un cost econòmic assumible que permet la seva utilització en un entorn real. Per plantejar un escenari per validar aquesta implementació, es necessita:

- Una blockchain Ethereum privada.

[9] URL: `https://github.com/ethereum/go-ethereum`.

- Un client amb unes credencials donades, que crea un comptador i simula un prestador de serveis que valida les credencials successives vegades fins esgotar-les.

S'ha utilitzat la implementació Go-Ethereum per desplegar la blockchain Ethereum privada. El programa client s'ha desenvolupat amb Golang. Per efectuar aquesta prova s'ha utilitzat un servidor amb un processador Intel i7 amb 4 nuclis a 2Ghz i 16GB de memòria RAM que executa la blockchain Ethereum privada i el programa client. El programa client utilitza l'API ethereum per Golang, i es comunica mitjançant sockets amb la blockchain, calcula cada ZKP necessària per cada pas i va cridant, primer l'SC de creació de comptador, i després, a cada iteració l'SC de consum del comptador, fins exhaurir-lo.

6.7.1 Resultats

La taula 6.3 resumeix el consum mitjà de gas de cada operació i per a cadascuna de les diferents funcions hash utilitzades en la seva implementació. Una fila per a cada operació i una columna per a cada funció hash (sha256 i keecak256).

Es pot veure en aquesta taula que els requisits en gas de la implementació amb keccak256 són una mica més baixos que la implementació amb sha256. S'ha de tenir en compte que Desplegament llibreries en aquesta taula es refereix al desplegament de la biblioteca BN256 comú en ambdues implementacions.

Operació	SHA256	KECCAK256
Desplegament llibreries	552986 Wei	552986 Wei
Desplegament Smart Contracts	2797421 Wei	2647239 Wei
Creació d'un comptador	610840 Wei	609024 Wei
Consum de credencials	76769 Wei	75542 Wei

Taula 6.3: Consum mitjà de gas

El cost monetari per la creació dels comptadors i el consum per ús dels mateixos és una mètrica clau per avaluar la viabilitat econòmica de la solució implementada, especialment quan es treballa amb una plataforma com Ethereum, on les comissions de transacció (anomenades "gas fees") poden afectar directament l'escalabilitat i l'eficiència d'un sistema.

En aquest cas, el cost per la creació d'un comptador inferior a un MWei (que equival a 10^{-12} Ethers) és extremadament baix. Aquesta xifra demostra que la creació d'un comptador en la tecnologia blockchain és pràcticament insignificant des del punt de vista econòmic, la qual cosa permet que múltiples comptadors es puguin crear sense representar una càrrega econòmica elevada per als usuaris o els operadors de la xarxa. Aquest baix cost és fonamental per garantir l'eficiència i l'accessibilitat del sistema, permetent-ne una adopció massiva i el seu ús en entorns on la creació de comptadors pot ser necessària a gran escala (per exemple, per a l'autenticació o el registre d'activitats d'usuaris en temps real).

D'altra banda, el consum per ús d'un comptador inferior a 100 KWei (equivalent a 10^{-13} Ethers) també és una quantitat extremadament petita. Aquest valor indica que cada acció que impliqui l'ús del comptador (com una transacció o una validació de credencial) requereix una quantitat mínima de recursos per a ser executada a la blockchain, garantint així que les

comissions de transacció no representaran una barrera econòmica per als usuaris, fins i tot en entorns amb un alt volum de transaccions.

Aquesta combinació de baixos costos de creació i ús fa que la solució sigui sostenible i escalable en entorns de producció, ja que permet una alta freqüència d'ús sense generar despeses excessives per a la infraestructura o per als usuaris finals. Això també millora la competitivitat de la solució, ja que les altres alternatives basades en tecnologia blockchain sovint poden generar costos significatius en funció de la seva estructura de comissions.

En resum, aquests costos molt baixos (inferiors a 1 MWei per creació i a 100 KWei per ús) són una característica positiva perquè garanteixen que la solució sigui assequible, escalable i pràctica en el context de la blockchain pública d'Ethereum, evitant que els costos de transacció limitin l'adopció de la solució en un entorn real.

6.8 Buscant els límits

Dins del procés d'avaluació de qualsevol SI s'ha de valorar el seu rendiment. En aquest cas, l'objectiu és verificar que Ethereum presenta un límit estructural que dificulta l'escalabilitat necessària per a la utilització massiva de comptadors basats en SC. Aquest límit, que en el cas d'Ethereum i amb els SCs desenvolupats, és de 40 Transaccions per Segon (TpS) representa una restricció significativa per a aplicacions que requereixen un alt volum de transaccions simultànies. Això es tradueix en una limitació en la capacitat d'escala del sistema, ja que si el nombre de transaccions supera aquest límit, la xarxa pot experimentar congestió, augmentant els temps de confirmació de transaccions i, en conseqüència, reduint l'eficiència global del sistema.

Aquesta limitació estructural és un factor crucial a tenir en compte quan es vol desplegar una solució basada en tecnologia blockchain per a casos d'ús que impliquin una gran quantitat de computacions, com seria el cas de comptadors que han de registrar múltiples esdeveniments o accions. En un sistema que depèn de la gestió i validació de les credencials de manera constant i en temps real, aquest límit podria generar retards, costos elevats en les comissions de transacció i una experiència d'usuari compromesa.

Per tant, tot i que Ethereum és una plataforma robusta i àmpliament utilitzada per a contractes intel·ligents, aquest límit de 40 TpS indica que pot no ser la millor opció per a solucions que requereixin una alta capacitat de transacció i un rendiment superior, com seria el cas dels comptadors basats en SC en un entorn de gran escala.

Per plantejar un escenari per validar aquesta limitació, es necessita:

- **Una blockchain privada Ethereum**: que serveix com a infraestructura d'emmagatzematge i execució dels comptadors. A diferència de l'apartat anterior, l'arquitectura d'aquesta xarxa és més complexa, consta de les següents peces:
  - **Un bootnode**: per l'arrancada i coordinació dels miners.
  - **Un monitor**: que és l'encarregat de recollir estadístiques.
  - **Un explorador**: que permet inspeccionar les transaccions.

  - **Un conjunt de miners**: depenent de l'experiment concret.
  - **Un balancejador de càrrega**: que des d'un únic punt d'entrada on es connecta el client, reparteix de manera rotatòria la càrrega a cada miner.

- **Els diferents usuaris**: que exerceixen el seu dret enviant una transacció a l'smart contract de comptador amb les seves credencials anònimes i l'identificador que han calculat.

Per implementar la xarxa privada Ethereum, s'ha optat per utilitzar el programari K3s[10] que proveeix la infraestructura per crear un cluster Kubernetes[11]. Per cada tipus d'arquitectura de xarxa, es desplega el Yet Another Markup Language (YAML)[12] de l'arquitectura que es vol provar, el qual crea la xarxa Ethereum desitjada. Cada peça de la xarxa Ethereum s'executarà en un contenidor dins del cluster Kubernetes. Un contenidor és com una màquina virtual independent.

S'ha modificat el programa client de l'Apartat 6.6.5 per poder fer noves proves tenint en compte múltiples usuaris simultanis per veure com això afecta el rendiment del sistema. El nou programa té com a paràmetres d'entrada Limit Gas Block (LGB), Block Generation Rate (BGR), TpS i un temps d'execució de la prova. Per simular usuaris simultanis utilitza diferents fils d'execució.

Per la infraestructura de la xarxa Ethereum, es tenen 3 ordinadors amb CPU Intel i7 amb 8 cores a 2.7 GHz, 16 MB de RAM i disc dur de 500 GB. Per la part client, es té 1 ordinador amb CPU Intel i7 amb 8 cores a 2.7 GHz, 16 GB de RAM i disc dur de 500 GB. Aquestes 4 màquines estan connectades a una xarxa de 1 Gbps.

Primer s'ha generat el material criptogràfic i les credencials per 200.000 usuaris. També s'han generat 8192 adreces Ethereum per poder invocar l'SC. Una mateixa adreça Ethereum pot actualitzar diversos comptadors ja que són les credencials les que identifiquen els comptadors. S'han realitzat les proves només pel cas de la nostra modificació.

Vist que en el cas que interessa, que és el cas d'utilització de credencials anònimes, el límit teòric de TpS és 8 per un BGR de 5 segons i 40 per un BGR d'1 segon, i com que el que es busca és portar el sistema al límit s'ha fixat el BGR a 1 segon que és el que dona un millor TpS.

El procediment de realització de les proves segueix els següents passos:

1. Per cada tipus d'arquitectura de xarxa, es desplega el fitxer YAML de l'arquitectura que es vol provar, el que crea la xarxa Ethereum desitjada.

2. S'executa el client que realitza l'experiment, tot enregistrant les mesures de temps d'enviament i validació per cada transacció. Aquest client realitza les següents accions:

   - Crea una comptador de 1.
   - Realitza el consum del comptador.

3. Repeteix el pas 2 variant la càrrega amb les premisses que s'expliquen seguidament.

[10] URL: `https://k3s.io/`.
[11] URL: `https://kubernetes.io/`.
[12] URL: `https://yaml.org/`.

4. Destrueix la xarxa.

S'han realitzat proves sobre el sistema variant els següents paràmetres:

- **P**: Temps durant el qual s'envien transaccions amb una càrrega donada. S'ha escollit 30 segons, 1 minut i 2 minuts.
- **TpS**: Número de transaccions per segon o càrrega a la que es sotmet el sistema durant el temps P. Es comença amb 30 transaccions per segon augmentant de 5 en 5, fins que la diferència entre el temps d'enviament de transaccions i el temps total (temps transcorregut des del primer enviament fins que es confirma la darrera transacció) superi més del 50% del temps de duració dels enviaments.
- **Arquitectura de la xarxa Ethereum**: s'ha volgut veure també la influència de nombre de miners, i s'ha provat 1, 2, 3 i 5.
- Mida del bloc: s'han realitzat les proves tant per la mida de bloc 30M com per la mida 60M.

6.8.1 Resultat

Autenticació	$TpB = \frac{LGB}{CGT}$	BGR	LGB	TpS	Màx. 12h
Sense comprovació	$\frac{30M}{108k} = 277$	1	30M	277	12M
		5		55	475.200
	$\frac{60M}{108k} = 555$	1	60M	555	24M
		5		111	1M
Credencials anònimes	$\frac{30M}{750k} = 40$	1	30M	40	1.7M
		5		8	69.120
	$\frac{60M}{750k} = 80$	1	60M	80	3.4M
		5		16	138.240

Taula 6.4: Límits teòrics de nombre de comptadors amb un únic punt d'enviament de transaccions i un LGB de 30 milions i 60 millions

S'ha realitzat proves sobre el sistema mesurant el següent (que correspon a les columnes de la Taula 6.5):

- $\overline{\Delta}$: mitjana del temps passat entre l'inici de l'enviament i la recepció de la confirmació de què la transacció s'ha integrat a la cadena.
- δ: desviació estàndard de la variable $\overline{\Delta}$.
- P_{real}: temps transcorregut des de l'inici de l'enviament fins a la recepció de la confirmació de la darrera transacció confirmada.

La Taula 6.5 es recull la mitjana del temps transcorregut entre l'enviament i la validació d'ús del comptador, així com la desviació estàndard per a cada cas. S'ha ressaltat en vermell la condició que marca l'aturada de la prova.

En la Figura 6.5 es pot veure la representació gràfica dels casos concrets BGR=1, LGB=30 milions, amb duracions de la prova de 2 minuts per configuracions de xarxa de 1, 2, 3 i 5 miners. A les abscisses hi ha el temps en què s'envia i a l'ordenada la mitjana del temps de confirmació de les transaccions llançades en aquell segon. S'ha dibuixat també, de color vermell en horitzontal, el temps de generació de bloc, per veure quant es dispersa el temps de validació d'aquest, i en vertical la durada de la prova.

6.8.2 Conclusions

De manera general, es veu que com més gran és el bloc, pitjors resultats; això és atribuïble a la falta de capacitat de procés de les màquines utilitzades. Per tant, no sembla bona idea intentar incrementar la capacitat augmentant la mida del bloc, encara que teòricament podria ser una opció.

Si es posa el focus en el nombre de miners, pel cas d'un sol miner, es veu que el límit real, 30 TpS, queda molt lluny del límit teòric, 40 TpS. Aquesta diferència és atribuïble a la càrrega que suposa per una única màquina haver de gestionar tantes peticions i minar alhora. En afegir més nodes, o sigui més capacitat de càlcul, es veu que es pot assolir el límit teòric amb més facilitat. Puntualment, pot absorbir càrregues majors, sacrificant temps d'espera, és el cas de 3 miners que pot arribar a 50 TpS durant 30 segons a una mitjana de 9 segons de temps de validació, però es pot veure que si s'allarga a un minut el temps es dispara. Això és possible gràcies a un sistema de cues que absorbeix temporalment la diferència entre el flux d'entrada (50 TpS) i el de sortida (límit teòric de 40 TpS per generació de blocs). Es veu també que en haver-hi més nodes, la càrrega es reparteix, però el temps de cada validació augmenta i això és degut a la necessitat de coordinació dels diferents nodes. També es pot observar que a mesura que s'acosta al límit, o l'ultrapassa, la desviació estàndard incrementa.

En la Figura 6.5 es mostra de manera gràfica i més detallada les dades de la Taula 6.5. Es representa, per un conjunt de transaccions llançades al mateix segon t_{env}, la mitjana del temps d'espera de la validació. Es pot comprovar que per càrregues per sota del límit teòric, qualsevol arquitectura de xarxa respon per sota dels 10 segons. Quan s'acosta al límit, 35 o 40 TpS, els temps es disparen i són més irregulars, i el comportament és similar per a totes les arquitectures; per tant, veiem que en arribar al límit teòric, la Blockchain es satura, i les transaccions han d'esperar cada cop més a ser validades.

6.9 Resum

En aquest capítol s'han detallat els resultats obtinguts i els experiments duts a terme durant la recerca. S'ha vist que no només s'han validat totes les peces, tant aïllades individualment com integrades i desplegades a la cadena, sinó que s'han pres mesures per determinar-ne la idoneïtat de la implementació. També s'ha dut al límit el sistema per fixar els seus límits, i veure que existeixen limitacions estructurals.

	30M					60M				
M	P	tps	$\overline{\Delta}$	δ	P_{real}	P	tps	$\overline{\Delta}$	δ	P_{real}
1	30s	30	2.94	0.18	31s	30s	30	2.93	0.26	31s
		35	43.07	8.47	1m3s		35	23.64	5.73	46s
	1m	30	2.89	0.2	1m1s	1m	30	2.58	0.53	1m1s
		35	45.46	10.76	1m55s		35	34.06	9.53	1m33s
	2m	30	2.38	0.49	2m1s	2m	30	2.91	0.3	2m2s
		35	91.85	31.84	3m49s		35	145.68	79.36	4m56s
2	30s	30	8.46	8.06	39s	30s	30	3.11	0.77	32s
		35	9.17	1.98	40s		35	8.75	5.07	35s
		40	11.91	4.64	44s		40	13.42	5.31	42s
		45	16.64	6.44	51s		45	29.3	10.7	53s
	1m	30	2.17	0.19	1m1s	1m	30	14.29	14.09	1m11s
		35	2.19	0.47	1m2s		35	14.71	8.38	1m17s
		40	16.48	8.98	1m25s		40	25.32	8.7	1m26s
		45	28.85	14.14	1m44s		45	40.7	18.23	1m59s
	2m	30	2.84	1.55	2m6s	2m	30	2.42	0.6	2m1s
		35	19.82	15.18	2m25s		35	3.26	1.16	2m2s
		40	47.26	24.94	3m12s		40	69.16	32.62	3m15s
3	30s	30	2.08	0.59	31s	30s	30	2.2	0.49	31s
		35	2.15	0.55	31s		35	5.86	4.29	35s
		40	5.84	1.35	35s		40	10.32	4.01	38s
		45	7.48	1.94	39s		45	11.05	4.99	42s
		50	9.02	3.86	44s		50	14.09	6.33	47s
		55	18.93	8.31	56s					
	1m	30	2.01	0.27	1m1s	1m	30	1.97	0.31	1m1s
		35	1.89	0.16	1m1s		35	2.02	0.24	1m1s
		40	3.63	0.96	1m4s		40	17.6	8.38	1m20s
		45	9	5.05	1m16s		45	28.01	10.46	1m35s
		50	17.66	10.21	1m35s					
	2m	30	2	0.18	2m1s	2m	30	1.92	0.15	2m1s
		35	2.17	0.21	2m1s		35	1.88	0.18	2m1s
		40	2.78	0.87	2m4s		40	24.88	17.92	2m37s
		45	25.56	20.64	2m54s		45	75.97	39.82	3m36s
		50	71.87	36.96	3m44s					
5	30s	30	5.76	3.38	37s	30s	30	3.24	1.06	32s
		35	9.88	7.55	43s		35	9.94	6.16	40s
		40	8.85	5.49	44s		40	17.6	10	53s
		45	15.22	7.33	52s					
	1m	30	2.56	0.62	1m2s	1m	30	2.13	0.46	1m1s
		35	7.35	7.29	1m11s		35	19.44	16.32	1m25s
		40	19.21	14.18	1m33s		40	34.6	15.02	1m45s
	2m	30	7.47	6.11	2m10s	2m	30	27.17	26	2m46s
		35	9.37	14.26	2m13s		35	44.38	33.76	3m7s
		40	41.8	29.43	3m14s					

Taula 6.5: Mitjana($\overline{\Delta}$) i desviació estàndard (δ) del temps transcorregut (Δ) entre l'enviament i la recepció de la validació. Variables: (1) durada de la prova (P), (2) TpS per un BGR d'un segon i per mides de bloc de 30 i 60 milions. P_{real} és la durada real de la prova.

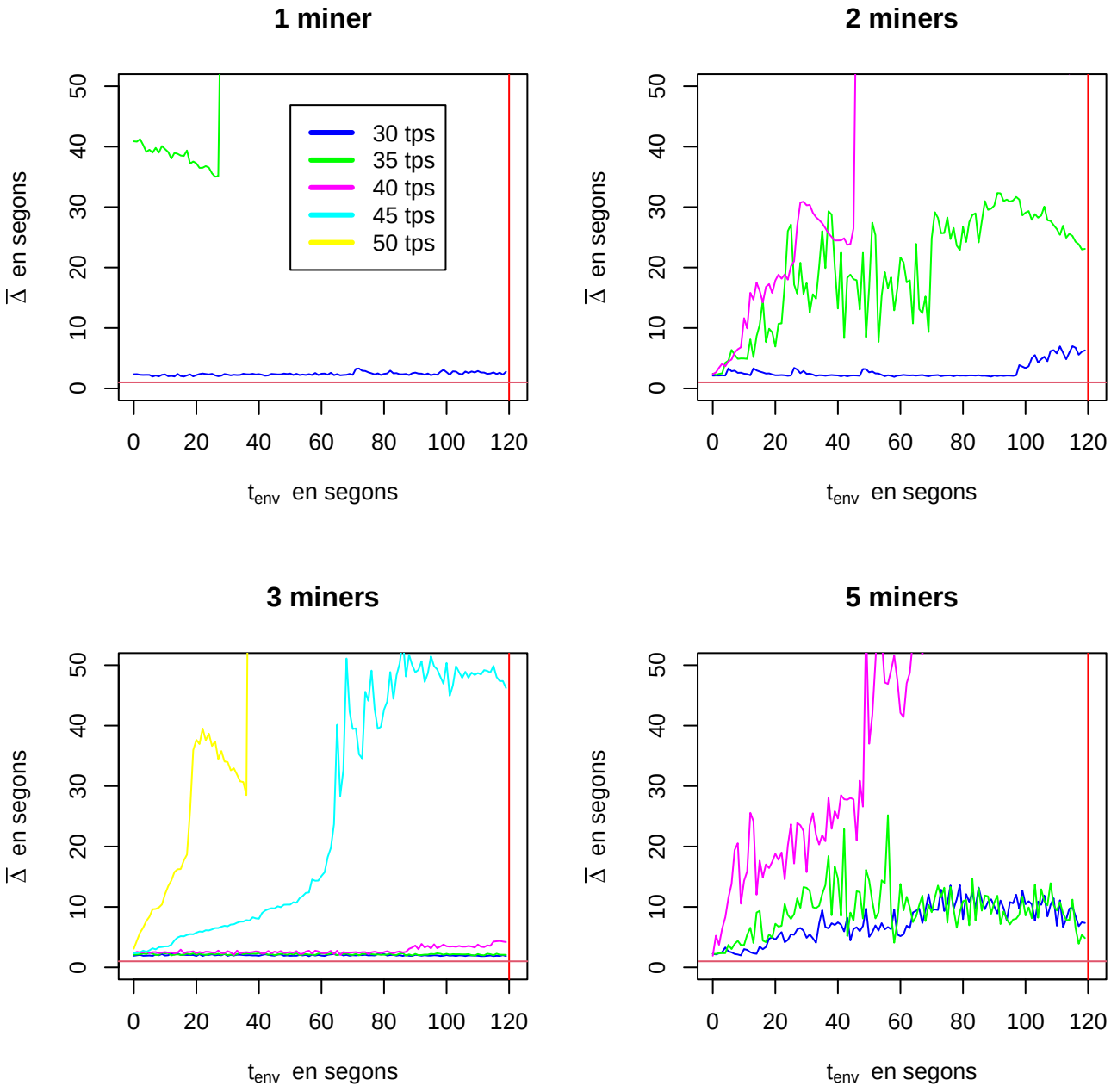

Figura 6.5: Resposta en el temps a diferents TpS amb BGR de 1 segon i LGB de 30 millions.

Capítol 7

Conclusions i futures línies de recerca

En aquest capítol, es presentaran les conclusions derivades de la recerca realitzada, així com les implicacions d'aquests en el context de la seguretat, la privadesa i l'escalabilitat en sistemes de verificació d'identitats digitals. S'ha abordat com les tecnologies emergents, com els pseudònims basats en atributs i els sistemes d'accés anònims amb usos limitats, poden contribuir a la creació de sistemes més segurs i respectuosos amb la privadesa.

A més de les conclusions generals, també es discutiran les futures línies de treball i investigació en aquest camp. Aquestes inclouen millores en la implementació de models d'anonimat en entorns descentralitzats, l'optimització de les solucions existents per a un millor rendiment i escalabilitat, així com l'exploració de noves tecnologies que puguin avançar en la creació d'identitats digitals segures, anònimes i flexibles. Les línies de treball futures seran essencials per afrontar els reptes emergents en matèria de seguretat i privadesa a mesura que les tecnologies continuïn evolucionant.

7.1 Conclusions

En aquest projecte de recerca s'ha aconseguit l'objectiu general de dissenyar i validar un sistema d'autenticació autogestionat amb credencials anònimes enllaçables. S'ha demostrat la seva validesa i seguretat, tal com es reflecteix en els resultats obtinguts. En aquest capítol es presenten les conclusions derivades de l'estudi realitzat, que s'agrupen en diferents àrees d'anàlisi: els objectius generals, l'anàlisi de seguretat i el rendiment del sistema.

L'objectiu general del projecte, que és "dissenyar i validar un protocol que permeti autoritzar un individu concret a realitzar una acció, un nombre de vegades fixat, preservant-ne el seu anonimat. A més, l'individu ha de poder exercir aquest dret lliurement, sense la necessitat de l'actuació de terceres parts de confiança", s'ha aconseguit amb èxit. S'ha proposat el disseny d'un protocol, s'ha implementat i s'ha demostrat la seva validesa mitjançant diversos experiments que han validat tant la funcionalitat com la seguretat del sistema.

L'objectiu general s'ha desglossat en diversos requisits, que han estat abordats de manera

independent, atès que cadascun representa un repte significatiu. S'han dut a terme modificacions a protocols ja existents, afegint nous paràmetres i funcionalitats. Aquestes modificacions s'han hagut de realitzar de manera controlada per garantir-ne la robustesa i validant la seva correcció.

La seguretat de la solució ha estat comprovada davant diversos escenaris maliciosos. El sistema es basa en la criptografia de corbes el·líptiques, els aparellaments bilineals i els problemes de logaritme discret, que han estat fonamentals per garantir la seva seguretat. A través d'una anàlisi exhaustiva, hem confirmat que el protocol d'autenticació implementat compleix amb els requisits següents:

- **Infalsificabilitat**: El protocol és invulnerable a falsificacions, garantint que només els usuaris autoritzats poden obtenir accés als serveis. Això vol dir que ningú pot fabricar unes credencials que es donin per vàlides si no ho son.
- **Anonimat de l'usuari**: El sistema protegeix la identitat de l'usuari, preservant la seva privadesa durant tot el procés d'autenticació. Es a dir, en cap moment ningú pot descobrir qui posseïx una determinada credencial si el propietari no vol. Presentar una credencial no proporciona cap informació sobre la identitat de qui presenta aquesta credencial.
- **No enllaçabilitat de l'identificador entre àmbits**: El sistema impedeix que els identificadors utilitzats en diferents àmbits siguin correlacionats. Dit d'altre manera, un individu que té dos identificadors en dos àmbits diferents es impossible trobar cap relació entre els dos identificadors.
- **Detecció de reutilització d'identificadors dins del mateix àmbit**: El protocol detecta qualsevol intent de reutilització d'identificadors. Aquesta propietat es la que permet implementar els comptadors d'ús.
- **Prevenint atacs de repetició**: S'inclouen mecanismes per prevenir l'ús no autoritzat de transaccions antigues mitjançant atacs de repetició. Això es, amb les proves de consentiment s'evita que algú malintencionat pugui reutilitzar unes credencials que se li han presentat.

Aquest conjunt de mesures fa que la solució sigui robusta i fiable, garantint tant la privadesa de l'usuari com la seguretat de la infraestructura. A més, aquest sistema podria aplicar-se en altres àmbits, com ara la votació electrònica limitant el comptador a un màxim de 1.

La seguretat del sistema ha estat una prioritat en aquest projecte. La solució proposada es fonamenta en mètodes criptogràfics avançats, com la criptografia de corbes el·líptiques i els aparellaments bilineals, que han demostrat ser eficients i segurs per protegir les credencials anònimes. L'anàlisi de seguretat realitzada ha confirmat que el sistema és capaç de resistir els intents d'atac més comuns, com els atacs de falsificació, de reutilització d'identificadors i de repetició.

Durant el desenvolupament, s'ha pogut constatar la conveniència d'utilitzar, sempre que sigui possible, tecnologies més obertes i àmpliament adoptades, com es va demostrar amb el pas de Hyperledger a Ethereum en la implementació final del protocol.

A l'hora de fer la implementació a Ethereum, sorgien dubtes sobre com la criptografia de corbes el·líptiques podria afectar el rendiment, especialment pel que fa als temps de càlcul. No obstant això, l'ús de contractes precompilats com els EIP-197 i EIP-196 ha permès optimitzar els processos criptogràfics, millorant considerablement l'eficiència. Aquesta optimització ha permès garantir que el sistema mantingués un bon rendiment fins i tot en entorns amb càrregues elevades, demostrant que la seguretat i el rendiment poden ser compatibles.

A més, l'anàlisi del rendiment ha revelat que la capacitat de la xarxa Ethereum té limitacions estructurals pel que fa a la quantitat de transaccions que pot processar per minut. A mesura que el nombre de miners augmenta, es van observant retards deguts a la congestió de la xarxa i la competència pels blocs. Per superar aquestes limitacions i millorar el rendiment, caldrà explorar solucions de Rollups o de capa 2, i una transició cap a nous sistemes de consens més eficients.

L'execució de codi a la blockchain implica un cost econòmic, ja que cada transacció requereix un cert consum de gas per validar-la. En aquest projecte, s'ha utilitzat un compte amb gas infinit per a realitzar les proves. No obstant això, és important considerar els costos econòmics associats a l'ús de la tecnologia blockchain, especialment en entorns de producció real.

Per a futures implementacions, es proposa la creació d'una blockchain Ethereum privada, on el cost de gas sigui nul, però mantenint els avantatges de transparència i fiabilitat de la tecnologia blockchain pública. Això permetria provar el sistema sense les limitacions econòmiques associades a les xarxes públiques. Tot i això, el model econòmic haurà de ser estudiat a fons per adaptar-se a un entorn de producció real.

Aquest projecte ha aconseguit desenvolupar un sistema d'autenticació anònima autogestionada, utilitzant una blockchain privada d'Ethereum. La solució proposada ha estat validada des del punt de vista de la seguretat i el rendiment, però també s'ha identificat la necessitat de millorar l'escalabilitat i l'eficiència econòmica en aplicacions a gran escala. En el següent capítol s'exploraran línies futures de recerca per millorar el sistema, especialment pel que fa al model econòmic i les possibles aplicacions en entorns de producció real.

7.2 Millores i futures línies de treball

Tot i que el sistema d'autenticació autogestionat amb credencials anònimes enllaçables desenvolupat durant aquest projecte ha estat validat amb èxit i demostra un alt nivell de seguretat i rendiment, sempre existeixen oportunitats de millora. A continuació es detallen algunes línies de treball futures que podrien optimitzar la solució i ampliar-ne l'aplicabilitat en diversos escenaris. Aquestes millores inclouen aspectes tècnics, econòmics i relacionats amb l'escalabilitat del sistema, així com possibles adaptacions a nous contextos d'ús.

7.2.1 Millores en l'escalabilitat i el rendiment

Una de les principals limitacions que es va identificar en l'actual disseny del sistema va ser la dependència de la xarxa Ethereum per a l'execució de les operacions de validació i autenticació. Tot i que s'ha aconseguit un bon rendiment amb l'ús dels contractes precompilats de EIP-197 i

EIP-196, hi ha diverses estratègies que podrien millorar l'escalabilitat i el rendiment global del sistema.

El sharding és una tècnica que permet dividir la xarxa de la blockchain en múltiples subxarxes (anomenades shards), cadascuna amb la seva pròpia capacitat de processar transaccions. Aquesta aproximació podria augmentar la capacitat de la xarxa Ethereum per a validar més transaccions per minut, millorant la velocitat i reduint la congestió de la xarxa. Integrar aquesta tècnica dins del sistema proposat podria accelerar els processos d'autenticació i millorar el temps de resposta en entorns amb un gran nombre d'usuaris.

Les tecnologies de segona capa com Plasma o els canals de pagament poden permetre a les transaccions realitzar-se fora de la cadena principal, per després ser validades de manera consolidada a la cadena Ethereum. Aquestes solucions podrien ajudar a alleugerir la càrrega de la xarxa Ethereum, permetent que les operacions d'autenticació es processin més ràpidament, i a un cost econòmic més baix.

Es podrien explorar maneres de fer més eficient el protocol d'autenticació. Per exemple, investigant la possibilitat d'utilitzar esquemes de criptografia més lleugers o protocols d'autenticació híbrids que combinin la seguretat de la criptografia de corbes el·líptiques amb altres tècniques criptogràfiques més ràpides, com els esquemes de signatura de curt termini. Aquestes optimitzacions podrien contribuir a una disminució dels temps de càlcul i una millor eficiència global del sistema.

7.2.2 Millores en la seguretat

Malgrat que la solució proposada és segura davant de diversos tipus d'atacs, la seguretat en l'àmbit de les tecnologies blockchain i criptogràfiques és un camp en constant evolució. Per tant, cal continuar investigant noves tècniques i mecanismes de seguretat per a reforçar la protecció del sistema.

Amb l'avenç de la computació quàntica, els algoritmes criptogràfics actuals poden veure's amenaçats en el futur. Per això, una línia de treball important seria la recerca d'alternatives criptogràfiques resistents a l'atac quàntic, com les criptografies basades en problemes matemàtics que siguin difícils de resoldre per computadors quàntics, com l'algoritme de l'efecte lattices. Aquesta adaptació garantiria que el sistema sigui segur davant de les futures amenaces que podrien sorgir amb l'evolució de la computació quàntica.

7.2.3 Millores econòmiques i models de micropagaments

En l'àmbit de la tecnologia blockchain, el cost econòmic associat a les transaccions és un factor limitant important. Tot i que el sistema ha estat dissenyat per funcionar sobre una blockchain Ethereum privada on el gas té cost nul, a la pràctica en una blockchain pública com Ethereum, el gas té un cost econòmic real que pot ser una barrera per a l'adopció massiva.

Una millora potencial seria desenvolupar un model econòmic més eficient per a l'ús de la tecnologia blockchain pública. Això inclouria explorar mecanismes per reduir el consum de gas per transacció, com la compressió de dades o l'ús de contractes intel·ligents més eficients. A més,

també s'hauria de considerar la implementació de sistemes de micropagaments que permetin als usuaris pagar de manera més flexible i econòmica per cada acció d'autenticació realitzada, sense que això impliqui un cost prohibitiu.

Una altra línia de treball seria la investigació de models de compensació econòmica per als usuaris que participen activament en el procés d'autenticació o validació, creant un sistema de recompenses o incentius basats en tokens. Això no només fomentaria la participació activa dels usuaris, sinó que també permetria generar un ecosistema econòmic al voltant del sistema d'autenticació, contribuint a la seva sostenibilitat a llarg termini.

7.2.4 Aplicacions en noves àrees d'ús

Les solucions d'autenticació anònima i autogestionada podrien ser aplicades en diversos altres àmbits, ampliant el seu ventall d'aplicacions.

Una de les possibles àrees d'aplicació és la votació electrònica. El sistema proposat permetria la creació de mecanismes de votació que asseguressin l'anonimat de l'elector i la validació d'un únic vot per persona. Aquesta aplicació podria ser útil en eleccions de diversos tipus, com eleccions polítiques o corporatives, on la seguretat, la privacitat i la confiança en el procés electoral són elements crucials.

En el sector de la salut, la privacitat de les dades és un aspecte fonamental. La solució d'autenticació anònima podria ser aplicada per garantir que els usuaris tinguin el control sobre qui accedeix a la seva informació mèdica, preservant-ne la privacitat i, al mateix temps, permetent un accés segur i validat a professionals autoritzats.

En altres àmbits, com la gestió d'identitats digitals per a serveis públics, l'ús d'aquest sistema podria permetre als ciutadans accedir a diversos serveis del govern de manera anònima, però segura, assegurant que només els usuaris autoritzats puguin realitzar certes accions o accedir a determinats recursos.

7.2.5 Propostes dels revisors anònims de la tesis

Com a línies futures, es preveu el desplegament del sistema en testnets públiques d'Ethereum com Sepolia o Goerli, amb l'objectiu de dur a terme proves més detallades que permetin validar els resultats obtinguts en aquest treball amb un major grau de realisme i rellevància. Aquest pas serà fonamental per a l'elaboració d'un estudi econòmic rigorós que avali la viabilitat del sistema en un entorn de producció. Paral·lelament, es contempla la possibilitat de realitzar una anàlisi energètica, tenint en compte tant el mecanisme de consens emprat com el tipus de maquinari utilitzat pels nodes participants.

En resum, tot i que el sistema desenvolupat és robust i eficaç, existeixen moltes oportunitats per millorar-lo, tant en termes de rendiment, seguretat com en l'eficiència econòmica. Aquestes línies de treball futures obren noves possibilitats per ampliar l'aplicabilitat del sistema a més àmbits i millorar la seva adopció a gran escala. La recerca contínua en aquestes àrees serà fonamental per garantir la seva evolució i adaptació a les noves demandes del món digital.

Referències bibliogràfiques

[1] A. Pfitzmann i M. Hansen. "A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management". A: *http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf* (ag. de 2010). (Cons. 25-01-2025) (v. les pàg. 5, 45).

[2] S. Goldwasser, S. Micali i C. Rackoff. "The Knowledge Complexity of Interactive Proof Systems". A: *SIAM Journal on Computing* 18.1 (1989), pàg. 186 - 208. DOI: 10.1137/0218012 (v. la pàg. 6).

[3] J. J. Quisquater et al. "How to explain zero-knowledge protocols to your children". A: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 435 LNCS. 1. Springer-Verlag, 1990, pàg. 628 - 631. ISBN: 9780387973173. DOI: 10.1007/0-387-34805-0_60 (v. la pàg. 6).

[4] H. Ong i C. P. Schnorr. "Fast Signature Generation with a Fiat Shamir — Like Scheme". A: *Advances in Cryptology — EUROCRYPT '90*. Ed. d'Ivan Bjerre Damgård. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pàg. 432 - 440. ISBN: 978-3-540-46877-6. DOI: 10.1007/3-540-46877-3_38 (v. la pàg. 8).

[5] D. Chaum i T. P. Pedersen. "Wallet databases with observers". A: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 740 LNCS (1993), pàg. 89 - 105. ISSN: 16113349. DOI: 10.1007/3-540-48071-4_7 (v. la pàg. 8).

[6] R. Cramer, I. Damgård i B. Schoenmakers. "Proofs of partial knowledge and simplified design of witness hiding protocols". A: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 839 LNCS (1994), pàg. 174 - 187. ISSN: 16113349. DOI: 10.1007/3-540-48658-5_19 (v. la pàg. 8).

[7] A. Fiat i A. Shamir. "How to prove yourself: Practical solutions to identification and signature problems". A: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 263 LNCS. 1987, pàg. 186 - 194. ISBN: 9783540180470. DOI: 10.1007/3-540-47721-7_12 (v. la pàg. 8).

[8] C. P. Schnorr. "Efficient signature generation by smart cards". A: *Journal of Cryptology* 4.3 (1991), pàg. 161 - 174. ISSN: 1432-1378. DOI: 10.1007/BF00196725 (v. la pàg. 8).

[9] J. H. Silverman. *Advanced Topics in the Arithmetic of Elliptic Curves*. New York: Springer New York, 2009. ISBN: 978-0-387-94325-1. DOI: 10.1007/978-1-4612-0851-8 (v. les pàg. 8, 11).

[10] F. Zhang, R. Safavi-Naini i W. Susilo. "An Efficient Signature Scheme from Bilinear Pairings and Its Applications". A: *Public Key Cryptography – PKC 2004*. Ed. de F. Bao, R. Deng i J. Zhou. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pàg. 277 - 290. ISBN: 978-3-540-24632-9. DOI: 10.1007/978-3-540-24632-9_20 (v. les pàg. 9, 11).

[11] D. Boneh i M. Franklin. "Identity-based encryption from the Weil pairing". A: *Annual international cryptology conference*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pàg. 213 - 229. DOI: 10.1007/3-540-44647-8_13 (v. la pàg. 10).

[12] D. Boneh i X. Boyen. "Efficient Identity-Based Encryption Without Random Oracles: IKE Protocol with Bilinear Pairings". A: *Proceedings of the 23rd Annual International Cryptology Conference (CRYPTO 2004)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pàg. 407 - 424. DOI: 10.1007/978-3-540-45146-4_24 (v. la pàg. 10).

[13] D. Galindo. "Boneh-Franklin Identity Based Encryption Revisited". A: *Automata, Languages and Programming*. Ed. de L.s Caires et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pàg. 791 - 802. ISBN: 978-3-540-31691-6. DOI: 10.1007/11523468_64 (v. la pàg. 10).

[14] K. Singh et al. "A novel credential protocol for protecting personal attributes in blockchain". A: *Computers and Electrical Engineering* 83.February (2020), pàg. 106586. ISSN: 00457906. DOI: 10.1016/j.compeleceng.2020.106586 (v. les pàg. 10, 17, 18, 35).

[15] L. Groth et al. "Short Signatures with Efficient Verification". A: *Proceedings of the 8th International Conference on Cryptography and Security (ACNS 2004)*. Berlin: Springer, 2004, pàg. 275 - 291. DOI: 10.1007/978-3-540-22668-7_21 (v. la pàg. 10).

[16] T. P. Pedersen. "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing". A: *Advances in Cryptology — CRYPTO '91*. Ed. de Joan Feigenbaum. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pàg. 129 - 140. ISBN: 978-3-540-46766-3. DOI: 10.1007/3-540-46766-1_9 (v. la pàg. 12).

[17] F. J. Corbató, M. Merwin-Daggett i R. C. Daley. "An experimental time-sharing system". A: *Proceedings of the May 1-3, 1962, Spring Joint Computer Conference*. AIEE-IRE '62 (Spring). San Francisco, California: Association for Computing Machinery, 1962, pàg. 335 - 344. ISBN: 9781450378758. DOI: 10.1145/1460833.1460871 (v. la pàg. 13).

[18] C. Weissman. "Security controls in the ADEPT-50 time-sharing system". A: *Proceedings of the November 18-20, 1969, fall joint computer conference*. 1969, pàg. 119 - 133. DOI: 10.1145/1478559.1478574 (v. la pàg. 13).

[19] D. Hardt. *The OAuth 2.0 Authorization Framework*. RFC 6749. RFC Editor, oct. de 2012. DOI: 10.17487/RFC6749. URL: http://www.rfc-editor.org/rfc/rfc6749.txt (v. la pàg. 14).

[20] *Information technology – Security techniques – A framework for identity management – Part 1: Terminology and concepts*. Standard ISO/IEC 24760-1:2011. 2011.

[21] International Organization for Standardization. *Information technology – Security techniques – A framework for identity management – Part 1: Terminology and concepts*. ISO/IEC 24760-1:2011. Vernier, Geneva, Switzerland: International Organization for Standardization, 2011 (v. la pàg. 14).

[22] ISO Central Secretary. *Information technology – Security techniques – A framework for identity management – Part 1: Terminology and concepts*. Standard. Geneva, CH: International Organization for Standardization, 2011.

[23] J. Camenisch i A. Lysyanskaya. "An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation". A: *Advances in Cryptology — EUROCRYPT 2001*. Ed. de B. Pfitzmann. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pàg. 93 - 118. ISBN: 978-3-540-44987-4. DOI: 10.1007/3-540-44987-6_7 (v. la pàg. 17).

[24] Security Team**, Computer Science Department, IBM Research. *Specification of the Identity Mixer Cryptographic Library*. Inf. tèc. Zurich, 8803 Rüschlikon, Switzerland, 2010, pàg. 1 - 52 (v. les pàg. 17, 18).

[25] S. Ringers, E. Verheul i J. Hoepman. "An Efficient Self-blindable Attribute-Based Credential Scheme". A: *Financial Cryptography and Data Security*. Ed. d'A. Kiayias. Cham: Springer International Publishing, 2017, pàg. 3 - 20. ISBN: 978-3-319-70972-7. DOI: 10.1007/978-3-319-70972-7_1 (v. la pàg. 18).

[26] G. Zyskind, O. Nathan i A. Pentland. "Decentralizing Privacy: Using Blockchain to Protect Personal Data". A: *2015 IEEE Security and Privacy Workshops*. 2015, pàg. 180 - 184. ISBN: 978-1-4799-9933-0. DOI: 10.1109/SPW.2015.27 (v. la pàg. 18).

[27] D. Johnson, A. Menezes i S. Vanstone. "The Elliptic Curve Digital Signature Algorithm (ECDSA)". A: *International Journal of Information Security* 1.1 (2001), pàg. 36 - 63. ISSN: 1615-5262. DOI: 10.1007/s102070100002 (v. la pàg. 18).

[28] R. L. Rivest, A. Shamir i L. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". A: *Communications of the ACM* 26.1 (gen. de 1983), pàg. 96 - 99. ISSN: 15577317. DOI: 10.1145/357980.358017 (v. la pàg. 18).

[29] B. Zundel et al. *Verifiable Credentials Data Model 1.0*. W3C Recommendation. W3C, nov. de 2019. URL: https://www.w3.org/TR/2019/REC-vc-data-model-20191119/ (cons. 31-01-2025) (v. les pàg. 19, 24).

[30] D. Reed et al. *Decentralized Identifiers (DIDs) v1.0*. W3C Working Draft. W3C, abr. de 2020. URL: https://www.w3.org/TR/2020/WD-did-core-20200421/ (cons. 31-01-2025) (v. les pàg. 21, 22).

[31] P. Saint-Andre i J. Klensin. *Uniform Resource Names (URNs)*. RFC 8141. RFC Editor, abr. de 2017. DOI: 10.17487/RFC8141. URL: http://www.rfc-editor.org/rfc/rfc8141.txt (v. la pàg. 22).

[32] D. Crockford i C. Morningstar. *Standard ECMA-404 The JSON Data Interchange Syntax*. Inf. tèc. Des. de 2017. DOI: 10.13140/RG.2.2.28181.14560 (v. la pàg. 22).

[33] D. Crockford. *The application/json Media Type for JavaScript Object Notation (JSON)*. RFC 4627. RFC Editor, jul. de 2006. DOI: 10.17487/RFC4627. URL: http://www.rfc-editor.org/rfc/rfc4627.txt (v. la pàg. 22).

[34] M. Van Wingerde. "BLOCKCHAIN-ENABLED SELF-SOVEREIGN IDENTITY - An exploratory study into the concept Self-Sovereign Identity and how blockchain technology can serve the fundamental basis Self-Sovereign Identity Management". Tesi doct. 2017. DOI: 10.13140/RG.2.2.17693.82406. URL: https://www.researchgate.net/publication/326914271 (v. la pàg. 26).

[35] William Mougayar. *The Business Blockchain*. 2016, pàg. 180. ISBN: 9781119300311.

[36] W. Mougayar i V. Buterin. *The Business Blockchain: Promise, Practice, and Application of the Next Internet Technology*. Wiley, 2016. ISBN: 9781119300335. URL: https://books.google.ad/books?id=CEsPDAAAQBAJ (v. la pàg. 27).

[37] D. Mills et al. "Distributed Ledger Technology in Payments, Clearing, and Settlement". A: *Finance and Economics Discussion Series* 2016.095 (2016), pàg. 971 - 1023. ISSN: 19362854. DOI: 10.17016/feds.2016.095 (v. la pàg. 27).

[38] J. De Kruijff i H. Weigand. "Understanding the blockchain using enterprise ontology". A: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10253 LNCS. 2017, pàg. 29 - 43. ISBN: 9783319595351. DOI: 10.1007/978-3-319-59536-8_3 (v. la pàg. 27).

[39] M. Swan. *Blockchain: Blueprint for a New Economy*. 2015, pàg. 149. ISBN: 9781491920497. DOI: 10.5555/3006358 (v. les pàg. 28, 31).

[40] NIST. "DRAFT FIPS PUB 202: SHA-3 standard: Permutation-based hash and extendable-output functions". A: *Federal Information Processing Standards Publication.* May (2015), pàg. 1 - 26. DOI: 10.6028/NIST.FIPS.202 (v. la pàg. 28).

[41] R. C. Merkle. "A Digital Signature Based on a Conventional Encryption Function". A: *Advances in Cryptology — CRYPTO '87*. Ed. de C. Pomerance. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pàg. 369 - 378. ISBN: 978-3-540-48184-3. DOI: 10.1007/3-540-48184-2_32 (v. la pàg. 28).

[42] A. Baliga. "Whitepaper — Understanding Blockchain Consensus Models". A: *Persistent* April (2017), pàg. 1 - 14 (v. la pàg. 31).

[43] A. Kiayias et al. "Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol". A: *Advances in Cryptology – CRYPTO 2017*. Ed. de J. Katz i H. Shacham. Cham: Springer International Publishing, 2017, pàg. 357 - 388. ISBN: 978-3-319-63688-7. DOI: 10.1007/978-3-319-63688-7_12 (v. la pàg. 32).

[44] M. Castro i B. Liskov. "Practical Byzantine Fault Tolerance and Proactive Recovery". A: *ACM Trans. Comput. Syst.* 20.4 (2002), pàg. 398 - 461. ISSN: 0734-2071. DOI: 10.1145/571637.571640 (v. la pàg. 32).

[45] S. De Angelis et al. "PBFT vs proof-of-authority: applying the CAP theorem to permissioned blockchain". A: *Italian Conference on Cyber Security (06/02/18)*. Gen. de 2018. URL: https://eprints.soton.ac.uk/415083/ (v. la pàg. 32).

[46] M. M. Islam, M. M. Merlec i H. P. In. "A Comparative Analysis of Proof-of-Authority Consensus Algorithms: Aura vs Clique". A: *2022 IEEE International Conference on Services Computing (SCC)*. 2022, pàg. 327 - 332. DOI: 10.1109/SCC55611.2022.00054 (v. la pàg. 32).

[47] L. Luu et al. "Making smart contracts smarter". A: *Proceedings of the ACM Conference on Computer and Communications Security*. Vol. 24-28-Oct. 2016, pàg. 254 - 269. ISBN: 9781450 341394. DOI: 10.1145/2976749.2978309 (v. la pàg. 33).

[48] N. Atzei, M. Bartoletti i T. Cimoli. "A survey of attacks on Ethereum smart contracts (SoK)". A: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10204 LNCS. 2017, pàg. 164 - 186. ISBN: 9783662544549. DOI: 10.1007/978-3-662-54455-6_8 (v. la pàg. 33).

[49] S. Azouvi, A. K. Parikh i P. Gohil. "Oracles: The Gateway between Blockchain and the Real World". A: *Proceedings of the 17th International Conference on Financial Cryptography and Data Security* (2018), pàg. 60 - 75. DOI: 10.1007/978-3-319-76468-6_5 (v. la pàg. 33).

[50] G. Wood et al. "Ethereum: A secure decentralised generalised transaction ledger". A: *Ethereum project yellow paper* 151.2014 (2014), pàg. 1 - 32 (v. les pàg. 33 - 35).

[51] G. Bertoni et al. "Keccak". A: *Advances in Cryptology – EUROCRYPT 2013*. Ed. de T. Johansson i P. Q. Nguyen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pàg. 313 - 314. ISBN: 978-3-642-38348-9. DOI: 10.1007/978-3-642-38348-9_19 (v. la pàg. 35).

[52] National Institute of Standards i Technology. "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions". A: Federal Information Processing Standards Publications (FIPS PUBS) 202 (2015). DOI: 10.6028/NIST.FIPS.202 (v. la pàg. 35).

[53] B.J. Oates. *Researching Information Systems and Computing*. SAGE Publications, 2006. ISBN: 9781412902243 (v. la pàg. 45).

[54] E. R. Verheul. "Self-Blindable Credential Certificates from the Weil Pairing". A: *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*. ASIACRYPT '01. Berlin, Heidelberg: Springer-Verlag, 2001, pàg. 533 - 551. ISBN: 3540429875. DOI: 10.1007/3-540-45682-1_31 (v. les pàg. 45, 51).

[55] F. Garcia-Grau, J. Herrera-Joancomartí i A. Dorca Josa. "Attribute Based Pseudonyms: Anonymous and Linkable Scoped Credentials". A: *Mathematics* 10.15 (2022). ISSN: 2227-7390. DOI: 10.3390/math10152548 (v. la pàg. 54).

[56] F. Garcia-Grau, J. Herrera-Joancomartí i A. Dorca Josa. "Anonymous Access System with Limited Number of Uses in a Trustless Environment". A: *Applied Sciences* 14.19 (2024). ISSN: 2076-3417. DOI: 10.3390/app14198581.

[57] K. Arnold, J. Gosling i D. Holmes. *Java(TM) Programming Language, The (4th Edition)*. Addison-Wesley Professional, 2005. ISBN: 0321349806 (v. la pàg. 70).

[58] N. Bitansky et al. "The hunting of the SNARK". A: *Journal of Cryptology* 30.4 (2017), pàg. 989 - 1066. DOI: 10.1007/s00145-016-9241-9 (v. la pàg. 80).

[59] R. Barbulescu i S. Duquesne. “Updating key size estimations for pairings”. A: *Journal of cryptology* 32 (2019), pàg. 1298 - 1336. DOI: 10.1007/s00145-018-9280-5.

[60] L. De Feo, D. Jao i J. Plût. “Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies”. A: *Post-Quantum Cryptography*. Ed. de B. Yang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pàg. 19 - 34. ISBN: 978-3-642-25405-5. DOI: 10.1007/978-3-642-25405-5_2.

[61] M. H. Nasir et al. “Scalable blockchains — A systematic review”. A: *Future Generation Computer Systems* 126 (gen. de 2022), pàg. 136 - 162. ISSN: 0167-739X. DOI: 10.1016/J.FUTURE.2021.07.035.

[62] Q. Zhou et al. “Solutions to Scalability of Blockchain: A Survey”. A: *IEEE Access* 8 (2020), pàg. 16440 - 16455. DOI: 10.1109/ACCESS.2020.2967218.

[63] S. Kim, Y. Kwon i S. Cho. “A Survey of Scalability Solutions on Blockchain”. A: *2018 International Conference on Information and Communication Technology Convergence (ICTC)*. 2018, pàg. 1204 - 1207. DOI: 10.1109/ICTC.2018.8539529.

[64] A. Hafid, A. S. Hafid i M. Samih. “Scaling Blockchains: A Comprehensive Survey”. A: *IEEE Access* 8 (2020), pàg. 125244 - 125262. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3007251.

[65] C. Li et al. “Scaling Nakamoto Consensus to Thousands of Transactions per Second”. A: (maig de 2018). DOI: 10.48550/arxiv.1805.03870.

[66] S. Goswami. “Scalability analysis of blockchains through blockchain simulation”. Treb. fin. de màst. University of Nevada, Las Vegas, 2017. DOI: 10.34917/10985898.

[67] W. Diffie i M. Hellman. “New directions in cryptography”. A: *IEEE Transactions on Information Theory* 22.6 (1976), pàg. 644 - 654. DOI: 10.1109/TIT.1976.1055638 (v. la pàg. 29).

Referències online

[68] A. Matyus. *Facebook Faces Another Huge Data Leak Affecting 267 Million Users — Digital Trends*. Des. de 2019. URL: https://www.digitaltrends.com/news/facebook-data-leak-267-million-users-affected/ (cons. 07-03-2020) (v. la pàg. 14).

[69] Google. *Google identity platform*. URL: https://developers.google.com/identity (cons. 25-01-2025) (v. la pàg. 14).

[70] Facebook. *Facebook login*. URL: https://developers.facebook.com/docs/facebook-login (cons. 25-01-2025) (v. la pàg. 14).

[71] OpenID. *OpenID Foundation website*. URL: https://openid.net/ (cons. 25-01-2025) (v. la pàg. 14).

[72] OASIS. *Security Assertion Markup Language (SAML) V2.0 Technical Overview*. URL: https://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html (cons. 25-01-2025) (v. la pàg. 14).

[73] C. Paquin i G. Zaverucha. *U-Prove Cryptographic Specification V1.1 (Revision 3)*. Des. de 2013. URL: https://www.microsoft.com/en-us/research/publication/u-prove-cryptographic-specification-v1-1-revision-3/ (cons. 25-01-2025) (v. les pàg. 17, 18).

[74] Moxy. *What is "Sovereign Source Authority"?* 2012. URL: https://www.moxytongue.com/2012/02/what-is-sovereign-source-authority.html (cons. 10-03-2020) (v. la pàg. 17).

[75] C. Allen. *The path to self-sovereign identity.* 2016. URL: http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html (cons. 30-01-2020) (v. la pàg. 17).

[76] W3C Techbology and society domain. *[EDITOR'S DRAFT] Verifiable Claims Working Group Frequently Asked Questions*. 2020. URL: https://w3c.github.io/webpayments-ig/VCTF/charter/faq.html#self-sovereign (cons. 25-01-2020) (v. la pàg. 23).

[77] CoinMarketCap. *Cryptocurrency Prices, Charts And Market Capitalizations*. URL: https://coinmarketcap.com/ (cons. 04-01-2023) (v. la pàg. 30).

[78] Investopedia. *Bitcoin Mining Definition*. Oct. de 2020. URL: https://www.investopedia.com/terms/b/bitcoin-mining.asp (cons. 02-01-2023) (v. la pàg. 30).

[79] Ethereum.org. *Proof of Stake FAQ*. Set. de 2024. URL: https://ethereum.org/ca/developers/docs/consensus-mechanisms/pos/ (cons. 19-02-2025) (v. les pàg. 31 - 33).

[80] Vitalic Buterin. *GProof of Stake FAQ*. 2020. URL: https://github.com/hyperledger/indy-sdk (cons. 08-01-2023).

[81] Ethereum.org. *Guia completa sobre Ethereum*. 2025. URL: https://ethereum.org/ca (cons. 02-01-2025) (v. la pàg. 33).

[82] Swarm. *Digital freedom now*. URL: https://www.ethswarm.org/ (cons. 04-02-2025) (v. la pàg. 33).

[83] Protocol Labs. *An open system to manage data without a central server — IPFS*. URL: https://ipfs.tech/ (cons. 04-02-2025) (v. la pàg. 33).

[84] B. Cohen. *The BitTorrent Protocol Specification*. 2008. URL: https://www.bittorrent.org/beps/bep_0003.html (cons. 04-02-2025) (v. la pàg. 33).

[85] Vitalik Buterin. *Ethereum white paper: a next generation smart contract & decentralized application platform*. 2013. URL: https://www.the-blockchain.com/docs/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf (cons. 22-02-2025) (v. les pàg. 34, 35).

[86] C. Reitwiessner. *EIP-196: Precompiled contracts for addition and scalar multiplication on the elliptic curve alt_bn128*. 2 de febr. de 2017. URL: https://eips.ethereum.org/EIPS/eip-196 (cons. 22-10-2022) (v. la pàg. 80).

[87] V. Buterin i C. Reitwiessner. *EIP-197: Precompiled contracts for optimal ate pairing check on the elliptic curve alt_bn128*. 6 de febr. de 2017. URL: https://eips.ethereum.org/EIPS/eip-197 (cons. 22-10-2022) (v. la pàg. 80).

[88] Blockchain.com. *Blockchain.com — Charts - blocks-size*. URL: https://www.blockchain.com/explorer/charts/blocks-size (cons. 22-02-2025).

[89] Aries-RFC. *Anoncreds Design*. 2020. URL: https://github.com/hyperledger-archives/indy-anoncreds (cons. 04-10-2020) (v. la pàg. 18).

[90] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. DOI: 10.2139/ssrn.3440802. URL: http://bitcoin.org/bitcoin.pdf (v. les pàg. 28, 31).

Publicacions

Durant la realització del doctorat s'han publicat dos articles. El primer article, titulat "Attribute Based Pseudonyms: Anonymous and Linkable Scoped Credentials", es va publicar al juliol de 2022 en el número 10 de l'edició especial Recent Advances in Security, Privacy, and Applied Cryptography" de la revista Mathematics, amb un Impact Factor de 2.3 (JCR) i un Citescore (Scopus) de 4 al 2024. Aquesta publicació té la referència 10(15)2548 i el DOI 10.3390/math10152548. És el resultat de la recerca realitzada sobre les credencials anònimes i l'$id_{\mathcal{U}}$, tractats als Apartats 5.1 i 5.2 del Capítol 5, respectivament.

L'article proposa un sistema de pseudònims basat en atributs (ABP) que permet proporcionar credencials anònimes però enllaçables en àmbits específics. Aquest sistema facilita l'emissió de credencials d'identitat digital que no només protegeixen la privacitat de l'usuari, sinó que també permeten vincular diverses accions d'una mateixa persona en un domini concret sense revelar la seva identitat. Aquest enfocament es distingeix per la seva aplicació en entorns de confiança mínima, com els sistemes de seguretat basats en criptografia avançada. La recerca descriu una metodologia robusta per a la creació i gestió d'aquestes credencials de manera que es manté l'anonimat mentre es possibilita l'enllaç de les accions realitzades.

El segon article, "Anonymous Access System with Limited Number of Uses in a Trustless Environment", va ser publicat al setembre de 2024 en el número 14 de l'edició especial Innovation in Information Security" de la revista Applied Sciences, amb un Impact Factor (JCR) de 2.5 i un Citescore (Scopus) de 5.3 al 2024. Aquesta publicació té la referència 14(19)8581 i el DOI 10.3390/app14198581. És el resultat de la recerca realitzada per a l'ampliació del protocol de credencials anònimes en un àmbit específic, amb l'objectiu d'incorporar comptadors anònims amb un nombre limitat d'accessos. Aquest tema es tracta als Apartats 5.4 i 5.3 del Capítol 5 i es presenta en el Capítol 6 de resultats, a l'Apartat 6.1.

L'article proposa un sistema d'accés anònim amb un nombre limitat d'usos en un entorn sense confiança. Aquest sistema fa servir una criptografia basada en proves de coneixement nul (ZKP), mitjançant les quals un usuari pot demostrar la seva autorització per a accedir a un servei sense revelar la seva identitat. A més, el sistema garanteix que un comptador d'usuari només es pot utilitzar un nombre limitat de vegades, el que ajuda a evitar l'ús fraudulent i la reutilització indeguda de les credencials.

El sistema també integra mesures de seguretat per protegir tant la privadesa de l'usuari com la integritat de l'entorn d'interacció, resolent així problemes comuns associats als sistemes d'autenticació en entorns de confiança mínima. L'article presenta una solució eficaç per assegurar que les credencials no siguin mal usades, mentre es manté l'anonimat i la seguretat de les

comunicacions.

Llistats

Aquest capítol conté els llistats referenciats en el document. S'han afegit agrupats per llenguatges de programació utilitzats.

7.3 Llenguatge Python

Llistat 7.1: Llibreries per cridar els serveis des de python

```
""#!/usr/bin/python3
import requests
import json

HOST="criptoserver"

def generateKeys():
    url = 'http://criptoserver:8000/user/generateKey'
    resp = requests.get(url)
    if resp.status_code != 200:
        # This means something went wrong.
        raise ApiError('GET /user/generatekey/ {}'.format(resp.status_code))
    # print('{}'.format(resp.json()));
    return resp.json();

def generateId(priv,scope,g2Pub):
    url = 'http://criptoserver:8000/user/generateId'

    data = dict();
    data['priv'] = priv
    data['scope'] = scope
    data['pub'] = g2Pub

    # print('{} {}'.format(url,json.dumps(data)));
    resp = requests.get(url,json=data)
    if resp.status_code != 200:
        # This means something went wrong.
        raise ApiError('GET /user/generateId/ {}'.format(resp.status_code))
    # print('{}'.format(resp.json()));
    return resp.json();

def generateKeyPairing():
    url = 'http://criptoserver:8000/user/generateKeyPairing'
    # print(url)
    resp = requests.get(url)
    if resp.status_code != 200:
        # This means something went wrong.
        raise ApiError('GET /user/generateKeyPairing/ {}'.format(resp.status_code))
```

```
    # print('{}'.format(resp.json()));
    return resp.json();

def commit(pub,scope,ver=None):
    url = 'http://criptoserver:8000/{}user/commitment'
            .format("" if ver == None else "{}/".format(ver))
    # print("****** COMMITMENT ****** {} {} {}".format(pub,scope,ver))
    data = dict();
    data['pub'] = pub
    data['scope'] = scope
    resp = requests.post(url,json=data)
    if resp.status_code != 200:
    # This means something went wrong.
        print('{}'.format(resp))
        raise ApiError('POST /user/commitment/ {}'.format(resp.status_code))
    return resp.json();

def generateZKP(secret,pub):
    url = 'http://criptoserver:8000/user/generateZKP'
    data = dict()
    data['secret'] = secret
    data['pub'] = pub
    resp = requests.post(url,json=data)
    if resp.status_code != 200:
    # This means something went wrong.
        raise ApiError('POST http://criptoserver:8000/user/generateZKP/ {}'
                       .format(resp.status_code))
    return resp.json()

def verifyZKP(A,t,pub,pubSecret):
    url = 'http://criptoserver:8000/ap/verifyZKP'
    data = dict()
    data['A'] = A
    data['t'] = t
    data['pub'] = pub
    data['pubSecret'] = pubSecret
    resp = requests.post(url,json=data)
    if resp.status_code != 200:
    # This means something went wrong.
        raise ApiError('POST /user/commitment/ {}'.format(resp.status_code))
    return resp.json()

def requestCertificate(commitment,zkpr,zkpage,pub,pubG2User,s,r,scope=None,ver=None):
    data = dict()
    data['commitment'] = commitment
    data['zkpr'] = zkpr
    data['zkpage'] = zkpage
    data['pub'] = pub
    data['pubG2User'] = pubG2User
    data['r'] = r
    data['s'] = s
    data['scope'] = scope
    data['ver'] = ver
    resp = requests.post('http://cpserver:5001/generateCertificate',json=data)
    if resp.status_code != 200:
    # This means something went wrong.
        raise ApiError('POST /user/commitment/ {}'.format(resp.status_code))
    return resp.json()
```

```
def generateCertificate(privAP,pubG2User,scope):
    url = 'http://criptoserver:8000/ap/generateCertificate'
    data = dict()
    data['privAP'] = privAP
    data['scope'] = scope
    data['pubG2User'] = pubG2User
    resp = requests.post(url,json=data)
    if resp.status_code != 200:
    # This means something went wrong.
        raise ApiError('POST /ap/generateCertificate {}'.format(resp.status_code))
    return resp.json()

def verifyCertificate(certificate,pubG1AP,pubG2User,scope):

    url = 'http://criptoserver:8000/user/verifyCertificate'
    data = dict()
    data['certificate'] = certificate
    data['pubG1AP'] = pubG1AP
    data['pubG2User'] = pubG2User
    data['scope'] = scope
    resp = requests.post(url,json=data)
    if resp.status_code != 200:
    # This means something went wrong.
        raise ApiError('POST /user/verifyCertificate/ {}'.format(resp.status_code))
    return resp.json()

def verifyIdu(scope,commitment,idu,pubG1User,pubG2User):

    url = 'http://criptoserver:8000/ap/verifyIdu'
    data = dict()
    data['commitment'] = commitment
    data['blindPubG1'] = pubG1User
    data['blindPubG2'] = pubG2User
    data['scope'] = scope
    data['idu'] = idu
    resp = requests.post(url,json=data)
    if resp.status_code != 200:
    # This means something went wrong.
        raise ApiError('POST /user/verifyIdu/ {}'.format(resp.status_code))
    return resp.json()


def blindCertificate(scope,certificate,pubG1AP,pubG1User,pubG2User):
    url = 'http://criptoserver:8000/user/blindCertificate'
    data = dict()
    data['scope'] = scope
    data['certificate'] = certificate
    data['pubG1AP'] = pubG1AP
    data['pubG1User'] = pubG1User
    data['pubG2User'] = pubG2User

    resp = requests.post(url,json=data)
    if resp.status_code != 200:
    # This means something went wrong.
        raise ApiError('POST /user/blindCertificate/ {}'.format(resp.status_code))
    return resp.json()

def verifyBlindCertificate(blindCommitment,blindCertificate,blindPubG1AP,
                           blindPubG1User,blindPubG2User,blindGenerator):
    url = 'http://criptoserver:8000/ap/verifyBlindCertificate'
```

```
    data = dict()
    data['blindCommitment'] = blindCommitment
    data['blindCertificate'] = blindCertificate
    data['blindPubG1AP'] = blindPubG1AP
    data['blindPubG1User'] = blindPubG1User
    data['blindPubG2User'] = blindPubG2User
    data['blindGenerator'] = blindGenerator

    resp = requests.post(url,json=data)
    if resp.status_code != 200:
    # This means something went wrong.
        raise ApiError('POST /ap/verifyBlindCertificate/ {}'.format(resp.status_code))
    return resp.json()
```

Llistat 7.2: Programa que verifica el Happy Path

```
""#!/usr/bin/python3
import lib
import json
import logging
import getpass

import os

logger = logging.getLogger('User')
logger.setLevel(logging.INFO)
ch = logging.StreamHandler()
ch.setLevel(logging.INFO)
formatter = logging.Formatter('%(asctime)s - %(filename)s - %(levelname)s - %(message)s')
formatter = logging.Formatter('%(asctime)s - %(filename)s'+
                              +'(%(lineno)s) - %(levelname)s - %(message)s')
ch.setFormatter(formatter)
logger.addHandler(ch)

keys = lib.generateKeys();
logger.info('KEYS -> pub: {}'.format(keys['pub']));
logger.info('KEYS -> pri: {}'.format(keys['priv']));
logger.info('KEYS -> invpri: {}'.format(keys['invpriv']));

keyPairing = lib.generateKeyPairing();
logger.info('KEYPAIRING -> pri: {}'.format(keyPairing['priv']));
logger.info('KEYPAIRING -> g1Pub: {}'.format(keyPairing['g1Pub']));
logger.info('KEYPAIRING -> g2Pub: {}'.format(keyPairing['g2Pub']));

commit = lib.commit(keys['pub'],21)
logger.info('COMMIT -> commitment : {}'.format(commit['commitment']))
logger.info('COMMIT -> random : {}'.format(commit['random']))

scope = "scope"
linkcommit = lib.linkcommit(keys['pub'],scope)
logger.info('LINKCOMMIT -> commitment : {}'.format(linkcommit['commitment']))
logger.info('LINKCOMMIT -> scope : {}'.format(linkcommit['scope']))

sign = lib.requestIvSign(commit['commitment'])
logger.info('SIGN -> r : {}'.format(sign['r']))
logger.info('SIGN -> s : {}'.format(sign['s']))

linksign = lib.requestIvSignLink(linkcommit['commitment'],scope)
logger.info('LINKSIGN -> r : {}'.format(linksign['r']))
logger.info('LINKSIGN -> s : {}'.format(linksign['s']))
```

```
scope = "scope"
id = lib.generateId(keys["priv"],linkcommit['commitment'],scope)
logger.info('ID -> id: {}'.format(id));
scope = "scope"

with open("iv.pub") as reader:
    ivPub = reader.read()

with open("g1CP.pub") as reader:
    pubG1CP = reader.read()

vs = lib.ivVerify(sign['s'],sign['r'],ivPub,commit['commitment'])
logger.info('VERIFY SIGNATURE -> r : {}'.format(sign['r']))
logger.info('VERIFY SIGNATURE -> s : {}'.format(sign['s']))
logger.info('VERIFY SIGNATURE -> commitment : {}'.format(commit['commitment']))
logger.info('VERIFY SIGNATURE -> ivPub : {}'.format(ivPub))
logger.info('VERIFY SIGNATURE -> {}'.format(vs['verify']))

linkvs=lib.ivLinkVerify(linksign['s'],linksign['r'],ivPub,linkcommit['commitment'],scope)
logger.info('VERIFY LINKSIGNATURE -> {}'.format(linkvs['verify']))

zkpr = lib.generateZKP(commit['random'],keys['pub'],'random')
logger.info('GENERATEZKP RANDOM ->{}'.format(json.dumps(zkpr)))

linkzkpr = lib.generateZKP(linkcommit['scope'],keys['pub'],'random')
logger.info('GENERATEZKP SCOPE ->{}'.format(json.dumps(linkzkpr)))

certificate = lib.requestCertificate(commit['commitment'],zkpr,zkpage,keys['pub'],
                                      keyPairing['g2Pub'],sign['s'],sign['r'])
logger.info('REQUESTCERTIFICATE ->{}'.format(json.dumps(certificate)))

linkcertificate = lib.requestLinkCertificate(linkcommit['commitment'],linkzkpr,linkzkpage,
                keys['pub'],c eyPairing['g2Pub'],linksign['s'],linksign['r'],scope)
logger.info('REQUEST LINKCERTIFICATE ->{}'.format(json.dumps(linkcertificate)))

vc = lib.verifyCertificate(commit['commitment'],certificate['certificate'],pubG1CP,
                           keyPairing['g2Pub'])
logger.info('VERIFY CERTIFICATE -> {}'.format(vc['verify']))

linkvc = lib.verifyCertificate(linkcommit['commitment'],linkcertificate['certificate'],
                               pubG1CP,keyPairing['g2Pub'])
logger.info('VERIFY LINK CERTIFICATE -> {}'.format(linkvc['verify']))

bCertificate = lib.blindCertificate(commit['commitment'],certificate['certificate'],
                                    pubG1CP,keyPairing['g2Pub'],keys['priv'],keys['pub'])
logger.info('BLINDCERTIFICATE -> {}'.format(json.dumps(bCertificate)))

linkbCertificate = lib.blindCertificate(linkcommit['commitment'],
                                        linkcertificate['certificate'],pubG1CP,
                                        keyPairing['g2Pub'],keys['priv'],keys['pub'])
logger.info('LINK BLINDCERTIFICATE -> {}'.format(json.dumps(linkbCertificate)))

vb = lib.verifyBlindCertificate(bCertificate['blindCommitment'],
                                bCertificate['blindPubG1CP'],
                                bCertificate['blindPubG2User'],
                                bCertificate['blindCertificate'],
                                bCertificate['blindGenerator'])
logger.info('VERIFY BLIND CERTIFICATE -> {}'.format(vc['verify']))
```

```
vb = lib.verifyBlindCertificate(linkbCertificate['blindCommitment'],
                                linkbCertificate['blindPubG1CP'],
                                linkbCertificate['blindPubG2User'],
                                linkbCertificate['blindCertificate'],
                                linkbCertificate['blindGenerator'])
logger.info('VERIFY BLIND LINK CERTIFICATE -> {}'.format(vc['verify']))

linkvb = lib.verifyBlindLinkCertificate(linkbCertificate['blindCommitment'],
                                        linkbCertificate['blindPubG1CP'],
                                        linkbCertificate['blindPubG2User'],
                                        linkbCertificate['blindCertificate'],
                                        linkbCertificate['blindGenerator'],id['id'],
                                        linkbCertificate['blindPrivUser'],scope)
logger.info('VERIFY BLIND LINK CERTIFICATE -> {}'.format(linkvb['verify']))
logger.info('VERIFY BLIND LINK CERTIFICATE -> {}'.format(id['id']))
linkvb = lib.verifyBlindLinkCertificate(bCertificate['blindCommitment'],
                                        bCertificate['blindPubG1CP'],
                                        bCertificate['blindPubG2User'],
                                        bCertificate['blindCertificate'],
                                        bCertificate['blindGenerator'],
                                        id['id'],bCertificate['blindPrivUser'],scope)
logger.info('VERIFY BLIND LINK CERTIFICATE -> {}'.format(linkvb['verify']))
```

Llistat 7.3: Intent de frau amb diferents àmbits

```
""#!/usr/bin/python3
import lib
import json
import logging
import getpass
import time
import os

logger = logging.getLogger('User')
logger.setLevel(logging.INFO)
ch = logging.StreamHandler()
ch.setLevel(logging.INFO)
formatter = logging.Formatter('%(asctime)s - %(filename)s - %(levelname)s - %(message)s')
formatter = logging.Formatter('%(asctime)s - %(filename)'+
                              's(%(lineno)s) - %(levelname)s - %(message)s')
ch.setFormatter(formatter)
logger.addHandler(ch)
starttime = time.process_time()
keys = lib.generateKeys();
logger.debug('KEYS -> pub: {}'.format(keys['pub']));
logger.debug('KEYS -> pri: {}'.format(keys['priv']));
logger.debug('KEYS -> invpri: {}'.format(keys['invpriv']));

keyPairing = lib.generateKeyPairing();
logger.debug('pub: {}'.format(keyPairing));
logger.debug('KEYPAIRING -> pri: {}'.format(keyPairing['priv']));
logger.debug('KEYPAIRING -> g1Pub: {}'.format(keyPairing['g1Pub']));
logger.debug('KEYPAIRING -> g2Pub: {}'.format(keyPairing['g2Pub']));

scope = "scope"
linkcommit = lib.linkcommit(keys['pub'],scope)
logger.debug('LINKCOMMIT -> commitment : {}'.format(linkcommit['commitment']))
logger.debug('LINKCOMMIT -> scope : {}'.format(linkcommit['scope']))

linksign = lib.requestIvSignLink(linkcommit['commitment'],scope)
```

```
logger.debug('LINKSIGN -> r : {}'.format(linksign['r']))
logger.debug('LINKSIGN -> s : {}'.format(linksign['s']))

id = lib.generateId(keys["priv"],linkcommit['commitment'],scope)
logger.debug('ID -> id: {}'.format(id));

with open("iv.pub") as reader:
    ivPub = reader.read()

with open("g1CP.pub") as reader:
    pubG1CP = reader.read()

linkvs=lib.ivLinkVerify(linksign['s'],linksign['r'],ivPub,linkcommit['commitment'],scope)
logger.debug('VERIFY LINKSIGNATURE -> {}'.format(linkvs['verify']))

linkzkpr = lib.generateZKP(linkcommit['scope'],keys['pub'],'random')
logger.debug('GENERATEZKP SCOPE ->{}'.format(json.dumps(linkzkpr)))

linkcertificate =lib.requestLinkCertificate(linkcommit['commitment'],linkzkpr,linkzkpage,
                                            keys['pub'],keyPairing['g2Pub'],linksign['s'],
                                            linksign['r'],scope)
logger.debug('REQUEST LINKCERTIFICATE ->{}'.format(json.dumps(linkcertificate)))

linkvc = lib.verifyCertificate(linkcommit['commitment'],linkcertificate['certificate'],
                               pubG1CP,keyPairing['g2Pub'])
logger.debug('VERIFY LINK CERTIFICATE -> {}'.format(linkvc['verify']))

linkbCertificate = lib.blindCertificate(linkcommit['commitment'],
                                        linkcertificate['certificate'],pubG1CP,
                                        keyPairing['g2Pub'],keys['priv'],keys['pub'])
logger.debug('LINK BLINDCERTIFICATE -> {}'.format(json.dumps(linkbCertificate)))
scope="scope1"
linkvb = lib.verifyBlindLinkCertificate(linkbCertificate['blindCommitment'],
                                        linkbCertificate['blindPubG1CP'],
                                        linkbCertificate['blindPubG2User'],
                                        linkbCertificate['blindCertificate'],
                                        linkbCertificate['blindGenerator'],
                                        id['id'],linkbCertificate['blindPrivUser'],scope)
logger.info('VERIFY BLIND LINK CERTIFICATE -> {}'.format(linkvb['verify']))
logger.info('VERIFY BLIND LINK CERTIFICATE -> {}'.format(id['id']))

logger.info('elapsed {}'.format(time.process_time() - starttime))
```

Llistat 7.4: Intent de frau amb diferents claus

```
""#!/usr/bin/python3
import lib
import json
import logging
import getpass
import time
import os

logger = logging.getLogger('User')
logger.setLevel(logging.INFO)
ch = logging.StreamHandler()
ch.setLevel(logging.INFO)
formatter = logging.Formatter('%(asctime)s-%(filename)s-%(levelname)s-%(message)s')
formatter = logging.Formatter('%(asctime)s-%(filename)s'+
                              '(%(lineno)s)-%(levelname)s-%(message)s')
```

```
ch.setFormatter(formatter)
logger.addHandler(ch)
starttime = time.process_time()
keysAlt = lib.generateKeys();
keys = lib.generateKeys();
logger.debug('KEYS->pub:{}'.format(keys['pub']));
logger.debug('KEYS->pri:{}'.format(keys['priv']));
logger.debug('KEYS->invpri:{}'.format(keys['invpriv']));

keyPairing = lib.generateKeyPairing();
logger.debug('KEYPAIRING->pri:{}'.format(keyPairing['priv']));
logger.debug('KEYPAIRING->g1Pub:{}'.format(keyPairing['g1Pub']));
logger.debug('KEYPAIRING->g2Pub:{}'.format(keyPairing['g2Pub']));

scope = "scope"
linkcommit = lib.linkcommit(keys['pub'],scope)
logger.debug('LINKCOMMIT->commitment:{}'.format(linkcommit['commitment']))
logger.debug('LINKCOMMIT->scope:{}'.format(linkcommit['scope']))

linksign = lib.requestIvSignLink(linkcommit['commitment'],scope)
logger.debug('LINKSIGN -> r : {}'.format(linksign['r']))
logger.debug('LINKSIGN -> s : {}'.format(linksign['s']))

id = lib.generateId(keys["priv"],linkcommit['commitment'],scope)
logger.debug('ID -> id: {}'.format(id));

with open("iv.pub") as reader:
    ivPub = reader.read()

with open("g1CP.pub") as reader:
    pubG1CP = reader.read()

linkvs=lib.ivLinkVerify(linksign['s'],linksign['r'],ivPub,linkcommit['commitment'],scope)
logger.debug('VERIFY LINKSIGNATURE -> {}'.format(linkvs['verify']))

linkzkpr = lib.generateZKP(linkcommit['scope'],keys['pub'],'random')
logger.debug('GENERATEZKP SCOPE ->{}'.format(json.dumps(linkzkpr)))

linkzkpage = lib.generateZKP("21",keys['pub'],'age')
logger.debug('GENERATEZKP LINK AGE -> {}'.format(json.dumps(linkzkpage)))

linkcertificate = lib.requestLinkCertificate(linkcommit['commitment'],linkzkpr,
                                              linkzkpage,keys['pub'],
                                              keyPairing['g2Pub'],
                                              linksign['s'],
                                              linksign['r'],scope)
logger.debug('REQUEST LINKCERTIFICATE ->{}'.format(json.dumps(linkcertificate)))

linkvc = lib.verifyCertificate(linkcommit['commitment'],linkcertificate['certificate'],
                               pubG1CP,keyPairing['g2Pub'])
logger.debug('VERIFY LINK CERTIFICATE -> {}'.format(linkvc['verify']))

linkbCertificate = lib.blindCertificate(linkcommit['commitment'],
                                        linkcertificate['certificate'],pubG1CP,
                                        keyPairing['g2Pub'],
                                        keysAlt['priv'],keysAlt['pub'])
logger.debug('LINK BLINDCERTIFICATE -> {}'.format(json.dumps(linkbCertificate)))

linkvb = lib.verifyBlindLinkCertificate(linkbCertificate['blindCommitment'],
                                        linkbCertificate['blindPubG1CP'],
```

```
                                        linkbCertificate['blindPubG2User'],
                                        linkbCertificate['blindCertificate'],
                                        linkbCertificate['blindGenerator'],id['id'],
                                        linkbCertificate['blindPrivUser'],scope)
logger.info('VERIFY BLIND LINK CERTIFICATE -> {}'.format(linkvb['verify']))
logger.info('VERIFY BLIND LINK CERTIFICATE -> {}'.format(id['id']))

logger.info('elapsed {}'.format(time.process_time() - starttime))
```

7.4 Llenguatge Solidity

Llistat 7.5: Llibreria BN256 amb funcions i tipus per treballar amb aparellaments

```
struct G1Point {
    uint X;
    uint Y;
}
// Encoding of field elements is: X[0] * z + X[1]
struct G2Point {
    uint Xx;
    uint Xy;
    uint Yx;
    uint Yy;
}

/// @return the generator of G1
function P1() internal pure returns (G1Point memory) {
    return G1Point(1, 2);
}

/// @return the generator of G2
function P2() internal pure returns (G2Point memory) {
    return G2Point(
      0x198e9393920d483a7260bfb731fb5d25f1aa493335a9e71297e485b7aef312c2,
      0x1800deef121f1e76426a00665e5c4479674322d4f75edadd46debd5cd992f6ed,
      0x090689d0585ff075ec9e99ad690c3395bc4b313370b38ef355acdadcd122975b,
      0x12c85ea5db8c6deb4aab71808dcb408fe3d1e7690c43d37b4ce6cc0166fa7daa);
}
/// @return the negation of p, i.e. p.add(p.negate()) should be zero.
function negate(G1Point memory p) internal pure returns (G1Point memory) {
  // The prime q in the base field F_q for G1
  uint q = 21888242871839275222246405745257275088696311157297823662689037894645226208583;
  if (p.X == 0 && p.Y == 0)
     return G1Point(0, 0);
  return G1Point(p.X, q - (p.Y % q));
}

function unmarshalG1(bytes memory enterBytes) public pure
returns (G1Point memory r)
{
  uint ax;
  uint ay;
  assembly {
      ax := mload(add(enterBytes,0x20))
      ay := mload(add(enterBytes,0x40))
  }
  r.X=ax;
  r.Y=ay;
```

```
  return (r);
}

function unmarshalG2(bytes memory enterBytes) public pure
returns (G2Point memory r)
{
  uint xx;
  uint xy;
  uint yx;
  uint yy;
  assembly {
      xx := mload(add(enterBytes,0x20))
      xy := mload(add(enterBytes,0x40))
      yx := mload(add(enterBytes,0x60))
      yy := mload(add(enterBytes,0x80))
  }
  r.Xx=xx;
  r.Xy=xy;
  r.Yx=yx;
  r.Yy=yy;
  return (r);
}

/// @return r the sum of two points of G1
function add(G1Point memory p1, G1Point memory p2) internal returns (G1Point memory r) {
    uint[4] memory input;
    input[0] = p1.X;
    input[1] = p1.Y;
    input[2] = p2.X;
    input[3] = p2.Y;
    bool success;
    assembly {
        success := call(not(0), 6, 0, input, 0xc0, r, 0x60)
    }
    require(success);
}
/// @return r the product of a point on G1 and a scalar, i.e.
/// p == p.mul(1) and p.add(p) == p.mul(2) for all points p.
function mul(G1Point memory p, uint s) internal returns (G1Point memory r) {
    uint[3] memory input;
    input[0] = p.X;
    input[1] = p.Y;
    input[2] = s;
    bool success;
    assembly {
        success := call(not(0), 7, 0, input, 0x80, r, 0x60)
    }
    require (success);
}

/// @return the result of computing the pairing check
/// e(p1[0], p2[0]) *  .... * e(p1[n], p2[n]) == 1
/// For example pairing([P1(), P1().negate()], [P2(), P2()]) should
/// return true.
function pairing(G1Point[] memory p1, G2Point[] memory p2) internal returns (bool) {
    require(p1.length == p2.length);
    uint elements = p1.length;
    uint inputSize = elements * 6;
    uint[] memory input = new uint[](inputSize);
    for (uint i = 0; i < elements; i++)
```

```
    {
        input[i * 6 + 0] = p1[i].X;
        input[i * 6 + 1] = p1[i].Y;
        input[i * 6 + 2] = p2[i].Xx;
        input[i * 6 + 3] = p2[i].Xy;
        input[i * 6 + 4] = p2[i].Yx;
        input[i * 6 + 5] = p2[i].Yy;
    }
    uint[1] memory out;
    bool success;
    assembly {
        success := call(not(0), 8, 0, add(input, 0x20), mul(inputSize, 0x20), out, 0x20)
    }
    require(success);
    require(out[0] != 0);
    return out[0] != 0;
}
/// Convenience method for a pairing check for two pairs.
function pairingProd2(G1Point memory a1, G2Point memory a2,
                      G1Point memory b1, G2Point memory b2) public
returns (bool) {
    G1Point[] memory p1 = new G1Point[](2);
    G2Point[] memory p2 = new G2Point[](2);
    p1[0] = a1;
    p1[1] = b1;
    p2[0] = a2;
    p2[1] = b2;
    return pairing(p1, p2);
}
```

Llistat 7.6: Funcions auxiliars de verificació del protocol.

```
function VerifyIdu(bytes memory scope, bytes memory commitment, bytes memory iduByte,
                   bytes memory g1Pub, bytes memory g2Pub) public
returns (bool res)
{
    uint256 h = uint256(sha256(scope));

    G1Point memory pubG1;
    pubG1 = unmarshalG1(g1Pub);

    G2Point memory pubG2;
    pubG2 = unmarshalG2(g2Pub);

    G2Point memory idu;
    idu = unmarshalG2(iduByte);

    G1Point memory C;
    C = unmarshalG1(commitment);

    G1Point memory leftG1;
    leftG1 = add(C,pubG1);
    leftG1 = mul(leftG1,h);

    res = pairingProd2(negate(leftG1),idu,P1(),pubG2);
    require(res);

    return res;
}
```

```
function VerifyBlindCertificate(bytes memory blindCommitment,
                                bytes memory blindCertificate,
                                bytes memory blindPubG1Byte,
                                bytes memory blindPubG2Byte,
                                bytes memory blindGenerator) public
returns (bool res)
  {
    G1Point memory bpubG1;
    bpubG1 = unmarshalG1(blindPubG1Byte);

    G1Point memory bgenG1;
    bgenG1 = unmarshalG1(blindGenerator);

    G2Point memory bpubG2;
    bpubG2 = unmarshalG2(blindPubG2Byte);

    G2Point memory bcertG2;
    bcertG2 = unmarshalG2(blindCertificate);

        //Compute b*H(C)*G1
    G1Point memory leftG1;
    leftG1 = unmarshalG1(blindCommitment);
          //Compute b*H(C) + b*pubG1CP)
          leftG1 = add(leftG1, bpubG1);

    res = pairingProd2(negate(leftG1),bcertG2,bgenG1,bpubG2);
    require(res);

    return res;
 }

function VerifyBlindCertificateAP(bytes memory blindPubG1Byte,
                                  bytes memory blindGenerator) public
returns (bool res)
  {
    G1Point memory bpubG1;
    bpubG1 = unmarshalG1(blindPubG1Byte);

    G2Point memory bgenG2;
    bgenG2 = unmarshalG2(blindGenerator);

    res = pairingProd2(negate(bpubG1),P2(),pubG1CP,bgenG2);
    require(res);

    return res;
}
```

Llistat 7.7: Verificació de la prova de consentiment

```
  // Parameters sku secret key and n value to increment
  // R = r*P;
  // t = H(R)*sku+n*r;
  function verifyZKPint(bytes memory R, uint256 t, uint n, bytes memory g1Pub)
   private returns (bool){
    Bn256.G1Point memory G;
    G = Bn256.P1();
    Bn256.G1Point memory pR;
    pR = Bn256.unmarshalG1(R);

    Bn256.G1Point memory pg1Pub;
```

```
    pg1Pub = Bn256.unmarshalG1(g1Pub);

    // Get t*P
    Bn256.G1Point memory pT = Bn256.mul(G,t);

    // Get c = H(R);
    //bytes32 b_c = sha256(abi.encode(R));
    uint256 H = uint256(sha256(R));

    // Get R*P and pku*h
    Bn256.G1Point memory npR = Bn256.mul(pR, n);
    Bn256.G1Point memory pkH = Bn256.mul(pg1Pub, H);

    // Add both points together
    Bn256.G1Point memory  pRight= Bn256.add(npR,pkH);

    // Verify. Do they match?
    if(pT.X == pRight.X && pT.Y == pRight.Y) {
        return true;
    } else {
        return false;
    }
  }
```

Llistat 7.8: Metodes de creació i consum del comptador.

```
function Create(bytes memory scope,bytes memory idu, bytes memory blindCommitment,
  bytes memory blindCertificate, bytes memory blindPubApG1,
  bytes memory blindPubUserG1, bytes memory blindPubUserG2,
  bytes memory blindGenerator, bytes memory blindGeneratorG2,
  uint max, uint256 t) public payable {
    require(VerifyIdu(scope, blindCommitment, idu, blindPubUserG1, blindPubUserG2),
            "VerifyIdu failed!");
    require(VerifyBlindCertificate(blindCommitment, blindCertificate, blindPubApG1,
                                   blindPubUserG2, blindGenerator),
            "VerifyBlindCertificatei failed!");
    require(VerifyBlindCertificateAP(blindPubApG1, blindGeneratorG2),
            "VerifyBlindCertificateAP failed!");
    bytes32 h=sha256(idc);
    require(verifyZKPint(abi.encodePacked(h),t,max+1,blindPubUserG1),
            "verifyZKPint failed!");
    uint idx=id2index[h];
    require(idx==0,"Counter already found!");
    totalt++;
    tCounter memory aux;
    aux.count=0;
    aux.max=max;
    counters[totalt] = aux;
    id2index[h] = totalt;
}

function Consume(bytes memory idc, uint n, uint256 t, bytes memory pubUser)
 public payable {
    require(verifyZKPint(idc,t,n,pubUser));
    bytes32 h=sha256(idc);
    uint idx=id2index[h];
    require(idx>0,"No counter found");
    tCounter memory aux = counters[idx];
    require(aux.count==n-1,"No valid proof");
    require(aux.count<aux.max,"Max usages reached");
```

```
    aux.count++;
    counters[idx] = aux;
}
```

7.5 Llenguatge Golang

Llistat 7.9: Structura de dades del wallet

```
type Wallet struct{
        Scope string `json:"scope"`
        G1AP string `json:"g1AP"`
        Keys KeyPairing `json:"keys"`
        Idu Idu `json:"idu"`
        Cred ABCred `json:"cred"`
}

type KeyPairing struct {
        Priv  string `json:"priv"`
        G1pub string `json:"g1pub"`
        G2pub string `json:"g2pub"`
}

type Idu struct {
          id string `json:"id"`
        G1 string `json:"idG1"`
        G2 string `json:"idG2"`
        C  string `json:"C"`
}

type ABCred struct {
        BC string `json:"bC"`
        Bcert string `json:"bCert"`
        Bg1Ap string `json:"bg1AP"`
        Bpriv string `json:"bpriv"`
        Bg1User string `json:"bg1User"`
        Bg2User string `json:"bg2User"`
        Bg1 string `json:"bG1"`
        Bg2 string `json:"bG2"`
}

type Counter struct{
        Idc string `json:"idc"`
        Zkp []*big.Int `json:"zkp"`
}
```