Butlletí de la Societat Catalana de Matemàtiques
Vol. 31, núm. 1, 2016. Pàg. 31–50. DOI: 10.2436/20.2002.01.65

Les matemàtiques al darrere de les criptomonedes

Elitza Maneva

Resum: Mitjançant una presentació dels fonaments de les criptomonedes com ara Bitcoin, exposarem tres temes que pertanyen a la criptografia moderna: les *firmes digitals*, les *funcions de hash* i les *demostracions de coneixement nul*. Els problemes de caràcter purament matemàtic que hi sorgeixen poden servir d'exemples als professionals docents per motivar l'estudi de l'aritmètica modular, la probabilitat, la teoria de grafs o la complexitat computacional.

Paraules clau: criptografia, criptomonedes, RSA, corbes el·líptiques, funció de *hash*, probabilitat, grafs, complexitat computacional, aritmètica modular.

Classificació MSC2010: 68-02, 68Q15, 68Q17, 68Q05.

1 Introducció

Fa sis anys que la moneda electrònica bitcoin atreu l'interès d'economistes, de polítics i de la ciutadania en general. Abans de Bitcoin, poca gent s'havia preguntat si la manera com funcionen els diners avui dia és l'única o la millor possible. Tot i que ningú no dirà que Bitcoin dóna respostes definitives a aquestes preguntes, clarament és una idea que obre moltes portes.

A part del fet de ser de codi obert, la novetat principal de Bitcoin és que és una moneda distribuïda, és a dir, no depèn de cap banc o autoritat central: ni per crear monedes, ni per efectuar transaccions, ni per assegurar la integritat de les monedes en circulació. Podria Bitcoin ser una oportunitat per canviar les regles del joc financer i fer-les més justes? Probablement és massa aviat per avaluar aquesta possibilitat. El que de ben segur és cert és que el fenomen Bitcoin representa una oportunitat excel·lent per explicar com les matemàtiques fan possibles coses que *a priori* semblen impossibles, com són, per exemple, firmar

Aquest article es basa en la lliçó inaugural del curs acadèmic 2014–2015 de la Facultat de Matemàtiques de la Universitat de Barcelona, impartida per l'autora. Una primera versió d'aquest text va aparèixer a les Publicacions de la Universitat de Barcelona.

documents sense ser físicament present en un lloc, regular el funcionament de sistemes distribuïts sense cap autoritat central, o demostrar que saps alguna cosa sense donar-ne més informació que el fet que la saps.

La criptografia ha entrat en gairebé tots els àmbits de la societat moderna, ja que és la branca de les matemàtiques que fa possible l'ús de xarxes públiques per a assumptes privats. Mitjançant Internet, aquesta ciència ha canviat la manera com ens relacionem i gestionem el dia a dia. Donades les moltes ocasions en què la seguretat d'Internet falla, segurament ja intuïu que aquesta ciència està molt poc desenvolupada. De fet, en bona part es basa en conjectures matemàtiques.

Podem, doncs, considerar que les criptomonedes són una especulació amb peus de fang? Hi ha molts arguments en contra de la seva adopció, però la confiança en la certesa de les conjectures matemàtiques és un dels més febles perquè, en realitat, tots els serveis electrònics dels bancs i la seguretat de l'ús de paraules clau secretes es basa en aquestes mateixes conjectures. L'argument més fort en contra de les criptomonedes és que al seu darrere no hi ha cap govern que en reguli el valor. En aquesta exposició no entrarem en qüestions econòmiques ni en comparacions amb les monedes tradicionals, conegudes per *monedes fiduciàries* (*fiat money*). Per entendre millor el protocol i el potencial del sistema és millor deixar de banda les preconcepcions i pensar *outside the box*.

2 Bitcoin

L'octubre de 2008 Satoshi Nakamoto (pseudònim d'una o de més persones anònimes) va enviar un article titulat «Bitcoin: A peer-to-peer electronic cash system» a un grup de notícies de recerca en criptografia [7]. Dos mesos més tard en va publicar també la implementació de programari (o codi) obert. Es basava en altres idees de monedes electròniques que no es van arribar a realitzar, tècniques clàssiques de la criptografia i el desenvolupament de xarxes d'igual a igual (*peer-to-peer*). La idea va agradar a molts criptògrafs i programadors en general. Alguns van contribuir millorant el codi. Molts més van instal·lar-lo i van començar a efectuar transaccions i a generar més monedes virtuals segons el protocol que explicarem a continuació.

Al principi, les transaccions eren intercanvis més o menys simbòlics, com, per exemple, dotacions de premis o pagaments a canvi de programari, i, de mitjana, cada 10 minuts es generaven 50 bitcoins més. La primera compra de veritat que es va efectuar amb bitcoins va ser un any i tres mesos més tard. Va ser la compra d'una pizza a canvi de 10 000 bitcoins. Això representava el 0.4 % de tots els bitcoins que existien en aquell moment. La cotització de 10 000 bitcoins quatre anys més tard va arribar als quatre milions d'euros.

A mesura que més gent va descarregar-se el codi i va començar a efectuar transaccions, el valor dels bitcoins (BTC) va començar a pujar. Es van crear diversos mercats i negocis al voltant de Bitcoin. Avui en dia hi ha milers de negocis que accepten pagament en bitcoins.

2.1 Protocol

El protocol és bastant senzill i les eines que fa servir són eines clàssiques de la criptografia.

Els participants estan organitzats en una xarxa d'igual a igual (*peer-to-peer*): cada participant es pot comunicar amb un nombre relativament petit d'altres participants, però de tal manera que, si bona part dels usuaris reenvia la informació que rep a totes les seves connexions, aquesta informació arriba a tothom.

Absolutament totes les transaccions es guarden en un fitxer, anomenat *cadena de blocs* (*block chain*). Aquest fitxer està disponible públicament i a més es guarda i s'actualitza regularment als ordinadors de tots els usuaris.[1] Consisteix en una seqüència de blocs; cadascun conté una sèrie de transaccions de bitcoins. A més de les transaccions, cada bloc conté una quantitat de bitcoins nous en concepte de premi. Per exemple, un bloc podria contenir la informació següent:

- La Marta envia 2 BTC al Jordi.
- L'Elsa envia 1.567 5566 BTC al Gerard.
- El Jordi rep 50 BTC nous.

És important subratllar que no hi ha cap autoritat que s'ocupi de mantenir la cadena de blocs. Tots els usuaris —qualsevol persona que s'hagi baixat el programari— col·laboren en el manteniment de la informació i tothom la guarda localment al seu ordinador. Si algú intenta manipular la història, no podrà fer-ho sol perquè la informació està replicada als ordinadors de tots els usuaris.

Si hom vol enviar una quantitat de bitcoins, tot el que ha de fer és anunciar-ho a les connexions pertinents. Aquestes connexions comproven si la persona té els bitcoins que diu que vol gastar mirant, a la cadena de blocs, totes les transaccions en què ha participat en el passat, i, si és així, llavors reenvien la transacció a les seves pròpies connexions. En el fons, un bitcoin és la seva pròpia història des que es va crear, totes les transaccions que l'afecten, i la darrera indica qui n'és el propietari.

En cada moment, qualsevol participant té constància d'una sèrie de transaccions entre usuaris que encara no apareixen a la cadena de blocs. Fent servir una idea anomenada *proof of work* o *demostració de feina* (vegeu la subsecció 4.2), aquesta sèrie de transaccions es reinterpreta com un trencaclosques, la resolució del qual és, de fet, un bloc que es pot afegir al final de la cadena de blocs. El participant que aconsegueix resoldre el seu trencaclosques abans de rebre una solució d'un altre participant, envia el bloc a totes les seves connexions, que en comproven la validesa, l'afegeixen al final de la seva còpia de la cadena de blocs i el reenvien a les seves connexions.

1 Aquesta és una petita simplificació. Encara que va ser així al principi, com que la cadena s'ha fet gran (més de 20 GB el setembre de 2014) i com que hi ha usuaris que fan servir dispositius petits, com ara mòbils, avui en dia molts clients no guarden tota la informació i només contribueixen a la connectivitat de la xarxa. La seguretat del protocol depèn dels usuaris amb la versió completa.

Els trencaclosques estan dissenyats perquè, de mitjana, se'n resolgui un cada 10 minuts entre tots els participants de la xarxa. Naturalment, l'autor de la resolució pot destinar els bitcoins nous en concepte de premi (l'última transacció en el bloc, la que no té cap originari) a si mateix o a qui vulgui. En tot cas, la quantitat de bitcoins que hi ha a la xarxa creix, de mitjana, cada 10 minuts, quan es genera un bloc nou, però el valor del premi és cada cop més baix, perquè es redueix a la meitat cada 210 000 blocs. L'any 2140 el valor del premi serà menys que un *satoshi*, que és la unitat mínima de bitcoins i equival a 10^{-8} BTC. Arribat aquest punt, en comptes de guanyar monedes noves, el guanyador del premi cobrarà només unes taxes voluntàries que s'especifiquen a cada transacció. Cada participant pot fer servir el seu criteri per incloure una transacció en el bloc en què està treballant o no, segons la taxa voluntària que s'especifiqui en aquesta transacció.

Per garantir que realment és la Marta qui autoritza la transacció «La Marta envia 2 BTC al Jordi» s'aplica una tècnica estàndard de la criptografia anomenada *criptografia de clau pública*. Aquesta tècnica ens proporciona una manera de firmar contractes de manera digital. Només si la frase està firmada per la Marta, la resta d'usuaris acceptaran la transacció com a autèntica. A la secció següent veurem un mètode concret per implementar firmes digitals basat en la teoria de números.

Els trencaclosques estan basats en les funcions de *hash* (conegudes per *funcions resum)*, que també es fan servir en el context de l'autenticació d'usuaris amb paraules clau, entre moltes altres aplicacions. Explicarem aquesta tècnica a la secció 4.

2.2 Evolució

El primer mercat de bitcoins va ser un lloc web per intercanviar cromos anomenat Mt.Gox (una abreviació de Magic: The Gathering Online Exchange). Va fer fallida a principis de 2014 per problemes tecnològics, però en aquell moment ja s'havien creat moltes alternatives més fiables.

Després de Bitcoin s'han creat desenes d'altres varietats de criptomonedes inspirades en el programari de Bitcoin, o en alguns casos còpies gairebé idèntiques a Bitcoin. Un exemple és Luckycoin, que es diferencia de Bitcoin pel fet que la quantitat del premi en cada bloc és aleatòria. Un altre és Dogecoin, que va començar a finals de 2013 com una broma (*doge* es refereix a una moda d'Internet sobre fotos de gossos en diverses situacions, amb els pensaments del gos escrits en idioma de gos: «Wow. Much funny.») i en sis mesos va arribar a tenir un valor total de 24 milions d'euros (comparat amb el valor de 5 000 milions d'euros del mercat de Bitcoin). Dogecoin, a més, és notable perquè se sol fer servir per a projectes altruistes com, per exemple, una campanya per ajudar a finançar l'equip de bob de Jamaica per anar als Jocs Olímpics de Sotxi i un altre per construir un pou a Kenya.

Fins ara només hi ha hagut un problema de seguretat en el programari de Bitcoin, trobat l'any 2010, i es va arreglar ràpidament. El problema més

greu de Bitcoin que fa que els criptògrafs busquin alternatives és la falta d'anonimat en el protocol [10]. Encara que els usuaris facin servir pseudònims —en general fins i tot fan servir pseudònims diferents per rebre diners de fonts diferents—, és molt fàcil fer servir la informació que hi ha a la cadena de blocs per identificar diverses entitats i persones i el flux de diners entre elles. Una alternativa que s'està implementant per abordar aquest problema és el protocol Zerocash [1], que fa servir proves de coneixement nul (*zero-knowledge proofs*). Ho veurem a la secció 5.

3 Firmes digitals

En el món físic verifiquem la identitat de les persones amb imatges: comparem la cara de la persona amb la foto a la targeta d'identitat o les corbes d'una firma amb les de l'original. En canvi, en el món digital tractem amb números. La firma no és més que un número que envia la persona, tradicionalment anomenada Alice, junt amb el missatge que vol firmar. El receptor, tradicionalment anomenat Bob, hauria de poder comprovar que aquest número només el pot haver generat l'Alice. En principi això no sembla possible perquè qualsevol persona, tradicionalment anomenada Òscar, d'*oponent*, pot intentar endevinar números fins que n'hi surti un que passi el test que fa servir el Bob per comprovar que el missatge ve de l'Alice. En el món físic això correspon a una falsificació de la firma. Però, de fet, falsificar una firma digital és bastant més difícil que falsificar una firma física, perquè si el test del Bob és complicat, pot ser que l'Òscar hagi de provar essencialment tots els números de la mida adequada per trobar el correcte. Si el número té una longitud de 100 dígits, per exemple, i l'Òscar triga un nanosegon per comprovar que un número passa el test del Bob, en total l'Òscar trigarà $\frac{10^{100}}{10^9\times 60\times 60\times 24\times 356} > 10^{82}$ anys per provar tots els números (l'edat de l'Univers és del voltant de 10^{10} anys).

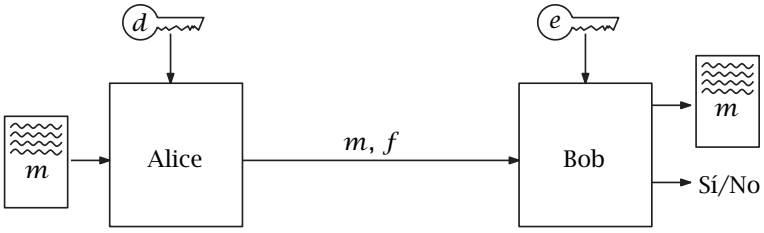

FIGURA 1: Protocol general de firmes digitals.

Una possibilitat que no es pot descartar és que, un cop coneguem els detalls del test que fa servir el Bob, se'ns faci molt més fàcil trobar un número que passi el test. Ben mirat, actualment les matemàtiques no disposen de tècniques prou fortes per demostrar que això sigui impossible per a algun d'aquests tests (o almenys no hem trobat cap test per al qual puguem demostrar-ho). Per

aquesta raó, la seguretat dels sistemes de firmes digitals encara es basa en conjectures sobre la dificultat d'alguns problemes computacionals. En aquest apartat en veurem dos exemples concrets.

Abans d'entrar en els detalls, és útil establir algunes notacions i introduir el concepte de *clau*. Hi ha dues claus que són dos números que fan servir l'Alice i el Bob durant el protocol. Una clau diem que és *privada*, perquè només la sap l'Alice, i la denotarem per d. L'altra és *pública*, perquè la pot saber qualsevol, i la denotarem per e. La clau privada es fa servir per crear el número que l'Alice envia junt amb el seu missatge m. D'aquest número se'n diu *firma* i el denotarem per f. La clau pública la genera l'Alice i l'envia al Bob (o a qualsevol altra persona) per poder comprovar que les firmes que rep de l'Alice són realment de l'Alice. Un esquema del protocol es pot veure a la figura 1.

3.1 Firmes i claus en el sistema Bitcoin

Per tenir bitcoins, l'usuari primer ha de crear una adreça (un número o una seqüència de caràcters, segons com vulgueu pensar-hi), en què es guardaran les monedes. Seria millor dir «a la qual s'associaran les monedes» perquè en realitat les monedes no tenen una realització física. Aquesta adreça, de fet, és la clau pública e del parell de claus (e, d) del protocol de firmes digitals. La clau privada d es guarda com un secret i es fa servir quan gastem els bitcoins per firmar transaccions. Cada transacció té una o més adreces d'origen i una o més adreces de destinació i s'ha de firmar amb les claus secretes de les adreces d'origen.

Per tant, si algú té la clau privada d'una adreça és com si tingués tots els bitcoins associats a aquesta adreça. Això és important perquè fa que els bitcoins siguin més aviat claus que no pas diners. Si un lladre fa una foto de la nostra clau és com si tingués la clau, però si fa una foto d'un bitllet que tenim és clar que no s'apropia d'aquest bitllet. Això ho van aprendre no fa gaire uns periodistes de la cadena de televisió Bloomberg dels Estats Units i van pagar la lliçó en bitcoins. Durant un reportatge sobre Bitcoin, van tenir l'ocurrència d'ensenyar l'imprès d'un regal de bitcoins que els havien donat. Amb la clau secreta al bell mig de la pantalla de la tele, en pocs segons aquests bitcoins van desaparèixer de la seva adreça.

3.2 Aritmètica modular

En la criptografia es fan servir números molt grans, per exemple, de centenars de dígits. Els protocols, com els de firmes digitals, demanen fer operacions amb aquests números, com ara, elevar un nombre de centenars de dígits a un altre d'una llargada comparable. El resultat és un número que no es podria guardar en un disc dur encara que estigués fet de tots els àtoms de l'Univers. Llavors, com fem aquests càlculs?

La idea és que, en comptes de fer servir l'aritmètica clàssica dels números enters, fem servir aritmètica modular. És a dir, en lloc de pensar en números

enters, pensem només en els residus que aquests números donen en dividir-los per un número especial N que escollim.

Recordeu que per denotar equivalències en l'aritmètica modular fem servir el símbol $\equiv$ en lloc del símbol $=$, i posem $(\text{mod } N)$ al final de l'equació per denotar que les equivalències són *mòdul* N. És a dir, tots els números que donen el mateix residu de divisió per N els considerem equivalents.

Les operacions de suma, resta i multiplicació mòdul un número gran es poden implementar de manera eficient fàcilment. El càlcul de l'invers mòdul un número gran també es pot implementar de manera eficient fent servir l'algoritme d'Euclides.

Finalment, queda per implementar l'operació d'exponenciació de manera eficient. Per calcular $a^b \pmod N$ de la manera òbvia hauríem d'executar b multiplicacions, calculant el residu de divisió per N cada vegada, per evitar que els resultats es facin llargs. El truc per fer aquest procés més ràpidament s'anomena *quadratura iterada* (*repeated squaring*). En comptes de calcular $a^2, a^3, a^4, \ldots, a^b$ calculem només $a^2, a^4, a^8, a^{16}, \ldots, a^{2^{\lfloor \log_2 b \rfloor}}$. Cada membre d'aquesta sèrie és el quadrat de l'anterior. El nombre de multiplicacions per generar aquesta sèrie és, doncs, $t = \lfloor \log_2 b \rfloor$, que és comparable al nombre de dígits de b (de l'ordre dels centenars, que són poques multiplicacions per a un ordinador).

Donada aquesta sèrie, podem calcular a^b fent només t multiplicacions més. Representem b com la suma d'un subconjunt dels números $\{1, 2, 4, 8, 16, \ldots, 2^{\lfloor \log_2 b \rfloor}\}$, és a dir,

$$b = \sum_{i=0}^{t} b_i \times 2^i,$$

per a alguns $b_0, b_1, \ldots, b_t \in \{0, 1\}$ (la representació binària de b).

Les t multiplicacions finals són:

$$a^b = a^{\sum_{i=0}^{t} b_i \times 2^i} = \prod_{i=0}^{t} a^{b_i \times 2^i} = \prod_{i \in \{0,\ldots,t\} : b_i = 1} a^{2^i}.$$

Per explicar el primer protocol de firmes digitals, que és RSA, necessitarem també el teorema d'Euler: per a cada N hi ha un número anomenat $\phi(N)$, tal que per a qualsevol x que no té divisors comuns amb N es compleix que

$$x^{\phi(N)} \equiv 1 \pmod N.$$

La funció d'Euler $\phi(N)$ és la quantitat de números més petits que N que no tenen divisors comuns amb N. Si N és un número primer, és clar que $\phi(N) = N - 1$. Si N és el producte de dos números primers p i q, com és el cas que ens interessa per entendre RSA, podem comprovar que

$$\phi(N) = (pq - 1) - (p - 1) - (q - 1) = (p - 1)(q - 1),$$

perquè, de tots els $pq-1$ números més petits que N, n'hi ha $p-1$ que són divisibles per q, $q-1$ que són divisibles per p i la resta no tenen divisors comuns amb N.

Necessitarem el corol·lari del teorema d'Euler següent.

PROPOSICIÓ 1. *Si N és producte de dos primers diferents p i q i α és un enter qualsevol, aleshores per a tot enter m es compleix*

$$m^{\phi(N)\alpha+1} \equiv m \pmod{N}.$$

PROVA. Hi ha tres casos:

1. Si $\mathrm{mcd}(m,N) = 1$, el teorema d'Euler ens diu que $m^{\phi(N)} \equiv 1 \pmod{N}$. Tenim
$$m^{\phi(N)\alpha+1} = (m^{\phi(N)})^{\alpha} \times m \equiv 1^{\alpha} \times m \equiv m \pmod{N}.$$
2. Si $\mathrm{mcd}(m,N) = N$, l'equivalència és certa perquè les dues bandes són 0.
3. Queda el cas que $\mathrm{mcd}(m,p) = p$ i $\mathrm{mcd}(m,q) = 1$ (o viceversa). És suficient demostrar que $m^{\phi(N)\alpha+1} \equiv m \pmod{p}$ i $m^{\phi(N)\alpha+1} \equiv m \pmod{q}$. La primera congruència és obvia, i la segona és conseqüència del teorema d'Euler com en el cas anterior, tenint en compte que $\phi(q) = q-1$,
$$m^{\phi(N)\alpha+1} = (m^{q-1})^{(p-1)\alpha} \times m \equiv 1^{(p-1)\alpha} \times m \equiv m \pmod{q}.$$

Amb aquestes eines per fer càlculs amb números grans en aritmètica modular, ja podem explicar el primer algoritme de firmes digitals.

3.3 Firmes mitjançant RSA

Un dels primers sistemes de firmes digitals va ser el de Ronald Rivest, Adi Shamir i Len Adleman de 1978, conegut per RSA [9]. Per crear les claus, l'Alice escull dos números primers p i q de molts dígits (centenars), que són secrets, i en calcula el producte

$$N = p \times q.$$

El producte N és informació pública, però no els números p i q. Una de les conjectures en què es basa el mètode és que si només tenim N, trobar p i q (*i. e.* factoritzar N) és un problema computacionalment difícil.[2]

Llavors, l'Alice escull dos números d i e, el d serà secret, el e, públic, tals que per a qualsevol missatge m el valor de $(m^d)^e$ mòdul N (el qual es pot calcular amb $2\log(d) + 2\log(e)$ multiplicacions de números de la mateixa mida que N) és el mateix que m. És a dir,

$$(m^d)^e \equiv m \pmod{N}.$$

2 Si un dia es construïssin ordinadors quàntics, amb aquests sí que podríem factoritzar números grans fent servir l'algoritme de Shor [11].

En el cas de RSA, només l'Alice pot calcular $\phi(N) = (p-1)(q-1)$ perquè només ella coneix p i q. Havent calculat $\phi(N)$, pot escollir e i d tals que

$$ed \equiv 1 \pmod{\phi(N)}.$$

La manera de fer-ho és trobar e que no tingui divisors comuns amb $\phi(N)$ (escollir e aleatòriament entre 1 i $\phi(N)-1$ funciona) i, mitjançant l'algoritme d'Euclides, calcular el seu invers mòdul $\phi(N)$, que serà d.

Un cop l'Alice té un parell de números amb aquesta propietat, el protocol és el següent: fent servir la clau secreta d, l'Alice calcula la firma

$$f = m^d \pmod{N}$$

i l'envia junt amb el missatge m.[3] El Bob agafa la firma i l'eleva a e mòdul N (recordeu que N és públic). Si el resultat és m, accepta la firma com a autèntica; si no, la rebutja.

Per convèncer-nos que el protocol és correcte només queda comprovar que $m^{de} \equiv m \pmod{N}$, el qual és conseqüència directa de la proposició 1.

Si l'Òscar vol firmar un altre missatge m fent veure que és l'Alice, necessitarà la clau d. La seguretat del protocol RSA depèn de la conjectura que és computacionalment difícil trobar d si només coneixes e i N (es pot comprovar que si pots factoritzar N, llavors sí que pots trobar d).

Cal mencionar que la mateixa idea es fa servir per xifrar i desxifrar missatges. El Bob pot xifrar els missatges que envia a l'Alice elevant-los a la clau pública e. És a dir, el Bob envia el missatge $\hat{m} = m^e \pmod{N}$. Només l'Alice pot desxifrar el missatge perquè té la seva clau privada d i pot calcular $\hat{m}^d \equiv \left((m^e)^d\right) \equiv m \pmod{N}$. D'aquí vénen els noms de les claus e i d: d'*encriptació* i *desencriptació*.

3.4 Firmes mitjançant ECDSA

El protocol de firmes digitals que fa servir Bitcoin no és RSA perquè els creadors van decidir optar per un protocol més modern i menys habitual però que comporta alguns avantatges com ara càlculs més eficients i claus més petites (per aconseguir la mateixa seguretat). És l'ECDSA o *Eliptic Curve Digital Signature Algorithm*, que va ser dissenyat els anys vuitanta ([5] i [6]), i es va popularitzar a principis d'aquest segle. La conjectura matemàtica en la qual es basa aquest protocol pertany a l'aritmètica dels grups associats a corbes el·líptiques sobre cossos finits. Concretament, es basa en la complexitat computacional del problema del logaritme discret.

3.4.1 Corbes el·líptiques El conjunt dels punts (x, y) que satisfan una equació no singular $y^2 = x^3 + ax + b$, a més d'un punt especial O que ens podem imaginar com l'infinit, és una corba el·líptica amb paràmetres a i b. Vegeu [12]

3 En realitat per calcular la firma, en els protocols de RSA i d'ECDSA, no es fa servir el missatge m original, sinó un *hash* del missatge $H(m)$ que té una llargada fixa. Fem servir m per comoditat. A la secció següent explicarem què és un *hash*.

per a la definició general de corba el·líptica.[4] Denotem aquest conjunt per $\mathcal{E}$. En el context de la criptografia, x i y són enters i l'aritmètica és la d'un cos finit, per exemple, la dels enters mòdul p per a algun número primer p. Per tant, l'equació que els punts satisfan és $y^2 \equiv x^3 + ax + b \pmod{p}$. Així, si $p = 23$, $a = 1$, $b = 0$, els punts de la corba el·líptica són

$$\begin{aligned}\mathcal{E} = \{&(0,0),(1,5),(1,18),(9,5),(9,18),(11,10),\\ &(11,13),(13,5),(13,18),(15,3),(15,20),(16,8),\\ &(16,15),(17,10),(17,13),(18,10),(18,13),(19,1),\\ &(19,22),(20,4),(20,19),(21,6),(21,17),O\}.\end{aligned}$$

Es pot definir una operació «$\odot$» sobre dos punts de la corba el·líptica de manera que $\mathcal{E}$ equipat amb aquesta operació sigui un grup abelià. El punt O en seria l'element identitat. Aquesta operació admet una interpretació geomètrica, il·lustrada a la figura 2. Es considera la recta que passa per P i Q i el tercer punt R de tall amb la corba. Aleshores $P \odot Q = -R$ on $-R$ és el simètric de R respecte l'eix de les x, és a dir, $P \odot Q = (r_x, -r_y)$ amb $(r_x, r_y) = R$. El punt $-R$ és l'oposat de R amb l'operació $\odot$ i es pot pensar com la tercera intersecció de la recta que passa per R i O amb la corba (les rectes per O són les rectes verticals).

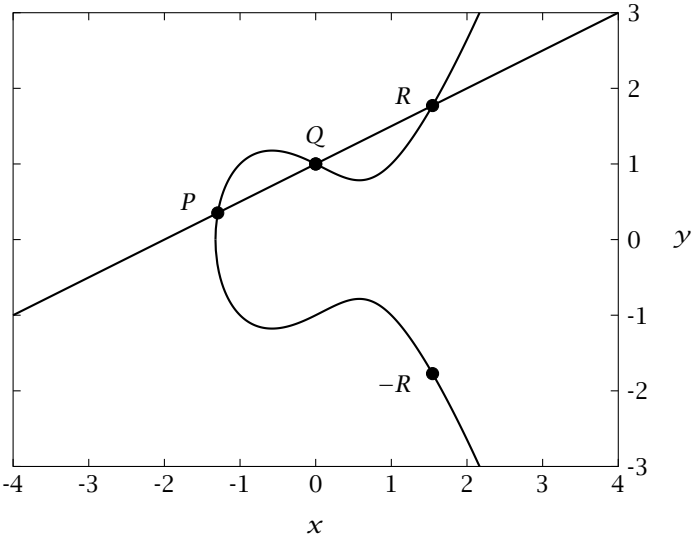

FIGURA 2: L'operació $\odot$ en una corba el·líptica amb equació $y^2 = x^3 - x + 1$.

En característica diferent de 2 i 3, es poden donar les coordenades de $P \odot Q$ de manera explícita. Suposem que $P = (p_x, p_y)$ i $Q = (q_x, q_y)$ són punts d'una corba el·líptica $\mathcal{E}$. Si $p_x = q_x$ i $p_y = -q_y$, llavors $P \odot Q = O$; en cas contrari,

4 Cal dir que en característica 2 o 3 no totes les corbes el·líptiques es poden reduir a l'expressió $y^2 = x^3 + ax + b$.

$P \odot Q = (r_x, r_y)$ on

$$\begin{aligned} r_x &:= s^2 - p_x - q_x, \\ r_y &:= (p_x - r_x)s - p_y \end{aligned}$$

per a

$$s := \begin{cases} (q_y - p_y)(q_x - p_x)^{-1}, & \text{si } P \neq Q, \\ (3p_x^2 + a)(2p_y)^{-1}, & \text{si } P = Q. \end{cases}$$

Farem servir la notació

$$S^t := \underbrace{S \odot S \odot S \odot \cdots \odot S}_{t}.$$

El problema del logaritme discret és: donats dos punts de la corba R i S, trobar un enter t tal que $S^t = R$. El t s'anomena *logaritme* en associació amb el logaritme continu i l'operació de multiplicació. El problema del logaritme discret, de fet, es pot plantejar a qualsevol grup. La seguretat de la criptografia de corbes el·líptiques es basa en la conjectura que trobar aquest t, donats R i S, és computacionalment difícil.

3.4.2 Protocol de firmes digitals El protocol té com a paràmetres (públics) la corba el·líptica definida per a, b i p, a més d'un element del grup amb ordre primer conegut. Diguem que aquest element és G i el seu ordre és n, és a dir, $G^n = O$.

La clau privada de l'Alice és un enter d escollit aleatòriament entre 0 i n. La seva clau pública és el punt $Q = G^d$. Per calcular-la l'Alice fa servir la quadratura iterada com quan elevem enters a una potència gran. El pas contrari, calcular d donat Q, és molt més difícil —concretament, és equivalent al problema del logaritme discret.

Donat un missatge m, l'Alice primer prepara un enter aleatori k secret que és nou per a cada nou missatge, s'assegura que k és coprimer amb n, i calcula $(p_x, p_y) = G^k$. Després fa servir la clau privada per calcular la firma, que en aquest cas té dues parts:

$$(f_1, f_2) := (p_x, (m + dp_x)k^{-1}) \pmod{n}.$$

Si f_1 o f_2 són zero, l'Alice tria un altre k.

El Bob rep el missatge m i la firma (f_1, f_2). Per verificar que la firma és autèntica el Bob calcula el punt

$$T = (t_x, t_y) := G^{mf_2^{-1}} \odot Q^{f_1 f_2^{-1}},$$

i comprova que $t_x = f_1$. Si la firma és vàlida el test surt correcte perquè $T = G^k$:

$$\begin{aligned} T &= G^{mf_2^{-1}} \odot Q^{f_1 f_2^{-1}} \\ &= G^{mf_2^{-1}} \odot G^{df_1 f_2^{-1}} \\ &= G^{(m+dp_x)f_2^{-1}} \\ &= G^k. \end{aligned}$$

4 Funcions de hash

L'altre ingredient principal en el disseny de Bitcoin és el concepte de *demostracions de feina* i, alhora, la creació de monedes noves. La tecnologia, en aquest cas, és molt senzilla i es basa en una idea ja força utilitzada: les funcions de hash.

Les funcions de *hash* tenen un munt d'aplicacions en el món digital. Una funció de hash pren d'entrada seqüències o fitxers de qualsevol mida i retorna seqüències de caràcters d'una mida fixa i petita, com ara 256 bits (o equivalentment 64 caràcters hexadecimals, 0–9 i a–f). Per il·lustrar-ho amb un exemple, imagineu-vos que dos amics que viuen lluny l'un de l'altre tenen fitxers que contenen la mateixa pel·lícula però no saben si els fitxers són idèntics. Per estar convençuts que ho són no cal que s'enviïn un dels fitxers sencer, que pot tenir una mida d'uns quants gigues. Els pot ser suficient aplicar una funció predeterminada, la funció de hash, que s'aplica al fitxer i dóna una línia de 256 bits, anomenada *el hash del fitxer*, que no té cap sentit o cap relació òbvia amb el contingut del fitxer. Tot i així, com que la funció és determinista, si els dos fitxers són idèntics, aquests 256 bits seran els mateixos. Les funcions de hash criptogràfiques normalment estan dissenyades de manera que si els dos fitxers es diferencien, encara que sigui només una mica, els dos hashos seran completament diferents en pràcticament tots el casos.[5] Per exemple, el hash de la frase de Douglas R. Hofstadter «If you think this sentence is confusing, then change one pig» és:

```
b0c840435c3ce497fc451f333ad18702443c1d822161150b8fa23bdb4ace97f9
```

i el hash de la mateixa frase amb punt final és:

```
380c4122fc5a3211bf801d8a02cc34928fecfc3732f1b4c6183d9ca608805b7d
```

De la mateixa manera, si els dos hashos són idèntics, podem estar pràcticament segurs que els dos fitxers són els mateixos fins a l'últim bit.

Encara que hi ha molts fitxers que tenen el mateix hash (perquè hi ha molts més fitxers possibles que seqüències de 256 bits), una bona funció de hash criptogràfica té la propietat que és molt difícil trobar dos fitxers amb el mateix hash. D'aquesta propietat se'n diu *resistència a col·lisions*. Una altra propietat d'aquestes funcions és que, si tenim el hash, és molt difícil recuperar el fitxer, o més ben dit, trobar algun fitxer que tingui aquest hash. Aquesta propietat s'anomena *resistència a inversions*.

Aquí ens trobem amb una altra conjectura. L'existència de funcions de hash que tenen aquestes dues propietats no està demostrada matemàticament. A la pràctica, al llarg dels anys s'han dissenyat diverses funcions de hash i algunes han estat «trencades» en el sentit que s'han trobat col·lisions o algoritmes

5 Hi ha funcions de hash per a altres aplicacions que tenen propietats diferents, per exemple, els locality-sensitive hashes, que funcionen exactament al revés: fitxers semblants tenen el mateix hash. En aquest article quan diem *hash* entendrem *hash criptogràfic*.

eficients per calcular-ne inversos. Actualment, la funció de hash més utilitzada es diu SHA-256 i és la que fa servir el protocol de Bitcoin.

Curiosament, les constants internes que fan servir les funcions de hash tenen explicacions senzilles, com per exemple els dígits del desenvolupament de π o els dígits de les arrels quadrades dels primers números primers. S'han escollit així per convèncer el públic que la funció no està dissenyada per cap agent maliciós, per exemple la NSA (National Security Agency), que tingui una clau secreta amb la qual pugui invertir la funció (cosa que igualment no es pot descartar del tot).

Els resultats d'aplicar una bona funció de hash a diferents fitxers són números que semblen generats a l'atzar. Si suposem que aquesta fos realment la distribució dels resultats, podem calcular la probabilitat de trobar-nos dos fitxers amb el mateix hash quan generem un gran nombre de hashos. Aquest experiment és conegut per *experiment dels aniversaris*. Imaginem-nos que els fitxers són persones i que els seus hashos són els seus aniversaris. A quantes persones necessitem preguntar-los l'aniversari per tenir una bona probabilitat de trobar-ne dues amb el mateix aniversari? Resulta que amb 23 persones ja tenim una probabilitat superior al 50 % d'èxit. D'aquest fet se'n diu *paradoxa dels aniversaris*.

Ens interessa el resultat del mateix experiment si l'any, en comptes de tenir 365 dies, en tingués molts més. Per posar un exemple, en el cas de la funció SHA-256, el hash del fitxer pot ser qualsevol número de 256 bits. D'aquests n'hi ha $2^{256} > 10^{76}$.

A continuació aproximem la probabilitat de col·lisió si tenim n dies diferents a l'any i m persones. Demostrarem que la probabilitat que dos aniversaris coincideixin passa de menys del 50 % a més del 50 % dins l'interval $m \in [\sqrt{n}, 2\sqrt{n}]$.

Denotem la probabilitat que hi hagi una col·lisió per $q(n,m)$ i la probabilitat que no n'hi hagi cap per $p(n,m) = 1 - q(n,m)$. Si les primeres k persones tenen aniversaris diferents, la probabilitat que l'aniversari de la persona següent a la qual ho preguntem coincideixi amb alguns dels k anteriors és k/n. Hi ha col·lisió si un d'aquests casos succeeix: la primera col·lisió es produeix o bé a l'hora de preguntar a la segona persona pel seu aniversari, o bé a l'hora de preguntar a la tercera, o bé a l'hora de preguntar a la quarta, etc. Per tant, la probabilitat que algun d'aquests casos es doni és la suma de probabilitats de tots els casos. D'altra banda, la probabilitat de tenir la primera coincidència en preguntar a la persona k-èsima és menor o igual que $(k-1)/n$. Per tant,

$$q(n,m) \leq \frac{1}{n} + \frac{2}{n} + \cdots + \frac{m-1}{n} = \frac{1}{n} \times \frac{(m-1)m}{2}.$$

Substituint m per $\sqrt{n}$ (suposant, per simplificar, que n és un quadrat perfecte), obtenim que:

$$q(n,\sqrt{n}) \leq \frac{(\sqrt{n}-1)\sqrt{n}}{2n} < \frac{1}{2}.$$

Per tant, si $m < \sqrt{n}$, la probabilitat de col·lisió encara està per sota del 50 %.

D'altra banda, podem fitar la probabilitat que no hi hagi cap col·lisió pensant en les últimes $m/2$ persones i exigint que els seus aniversaris siguin diferents dels de les primeres $m/2$ (suposant, per simplificar, que m és parell). Per tant,

$$p(n,m) \leq \left(\frac{n-m/2}{n}\right)^{m/2}.$$

Posem $m = 2\sqrt{n}$ i tenim

$$p(n,2\sqrt{n}) \leq \left(1-\frac{1}{\sqrt{n}}\right)^{\sqrt{n}} < e^{-1} < \frac{1}{2}.$$

Per tant, si $m \geq 2\sqrt{n}$ es compleix $p(n,m) < 1/2$ i $q(n,m) > 1/2$.

En el cas de SHA-256, $n > 10^{76}$, i per tant per tenir una probabilitat del 50% de veure dos fitxers amb el mateix hash hem de provar més de 10^{38} fitxers. Això fa que sigui pràcticament impossible trobar dos fitxers amb el mateix hash fent servir aquest mètode.

4.1 Paraules clau

Una de les aplicacions més importants de les funcions de hash és el seu ús per guardar paraules clau d'usuaris. La bona pràctica és evitar guardar les paraules clau tal com són, en un fitxer, perquè hi ha el perill que algú el pugui obrir. La manera de protegir-se d'aquest atac és la següent: en comptes de guardar les paraules clau, se'n poden guardar els hashos. Quan l'usuari envia la paraula clau c al servidor, aquest aplica la funció de hash H i compara el resultat $H(c)$ amb el hash que té guardat en el fitxer de hashos. D'aquesta manera encara que algú obtingués el fitxer, igualment no hi tindria accés perquè no podria recuperar les paraules que generen aquests hashos.

Si donem el disseny per fet, un atacant podria generar un diccionari de moltes de les possibles paraules clau (típicament combinacions d'uns vuit caràcters) junt amb els seus hashos i consultar aquest diccionari per recuperar les paraules clau, donats els seus hashos. Per evitar aquest atac, junt amb el hash es guarda un número aleatori s (relativament gran), anomenat *sal*, i el hash es calcula sobre la paraula clau concatenada amb la sal.

Finalment, per evitar que algú pugui fer moltes proves amb paraules clau de manera automàtica, la funció de hash s'aplica recursivament unes quantes vegades. En lloc de $H(c+s)$ es guarda $H(H(H\ldots H(c+s)\ldots))$, amb la funció de hash aplicada unes mil vegades o més per tal que la comprovació trigui un temps que no serà notable per a un usuari honest però que pot impedir un atacant automàtic.

4.2 Demostracions de feina en Bitcoin

El que fa Bitcoin especial és un mètode enginyós per aconseguir que la verificació de les transaccions sigui una tasca distribuïda, compartida entre tots els participants, que no s'hagin de refiar d'un banc central, i crear incentius

per col·laborar en aquesta tasca. La tècnica per aconseguir-ho es basa en les funcions de hash.

Com ja hem explicat a la introducció, el primer que fa un usuari quan rep una transacció és verificar-la. Tot i així, una transacció de bitcoins es converteix en oficial només quan la transacció s'afegeix a la cadena de blocs com a part d'un bloc. El procés de creació de blocs, el qual anomenem *trencaclosques*, és una tasca d'una gran complexitat computacional, que expliquem a continuació.

La tasca consisteix a trobar una seqüència de caràcters tal que, si adjuntem el contingut del bloc (la sèrie de transaccions) a aquesta seqüència i apliquem la funció de hash SHA-256, el resultat és un número més petit que un número donat: *l'objectiu* (*target*). La mida de l'objectiu determina la dificultat del problema i es regula cada dues setmanes per tal que algú a la xarxa trobi una solució, de mitjana, cada 10 minuts. L'objectiu és cada cop més petit perquè la tecnologia dels processadors d'ordinadors millora, i cada cop hi ha més usuaris i més recursos computacionals empleats a la tasca.

Com veieu, la tasca és molt semblant a la inversió de hashos. Per tant, el més probable és que ningú no tingui cap altra manera de resoldre'l que no sigui provant múltiples seqüències aleatòries de caràcters fins que una d'elles tingui un hash amb la propietat desitjada. El procés per resoldre aquests trencaclosques computacionals es coneix per *mineria* (*mining*), perquè amb cada bloc es creen bitcoins nous (el premi) i recorda la tasca de buscar or en una mina (una tasca bastant feixuga i inútil per si sola).

Per què cal que la tasca de crear blocs nous sigui computacionalment difícil? La idea és que d'aquesta manera s'evita que un participant domini el procés de verificació i manipuli la cadena de blocs, per exemple incloent-hi dues transaccions diferents que fan servir els mateixos bitcoins (un atac conegut per *doublespending*). La idea no és totalment nova i es coneix per *demostració de feina* (*proof of work*). Originalment es va inventar per combatre el correu brossa (*spam*) [2]. En aquest context, per enviar un correu, primer es demana al servidor que l'envia que resolgui un problema computacional que normalment requereix uns segons; així s'evita que algú pugui enviar un gran nombre de correus en molt poc temps.

Sovint passa que dos blocs, tots dos vàlids, es creen aproximadament al mateix temps. Llavors hi ha una bifurcació en la cadena de blocs i alguns participants treballen per continuar una cadena mentre que d'altres treballen per continuar-ne l'altra. El protocol exigeix que es continuï la que sigui més llarga. Normalment, en pocs minuts (suficients per crear tres o quatre blocs nous) una de les dues branques guanya l'altra en longitud i ja es pot considerar que és la cadena oficial perquè la segona mai no arribarà a atrapar la primera.

5 Tecnologies criptogràfiques més avançades

Com ja hem destacat a la introducció, és important pensar en Bitcoin com el primer pas important (o almenys el primer pas d'impacte real) cap al desenvolupament de protocols distribuïts de caràcter econòmic o financer. Aquest és

un tema de recerca actiu i no pas una qüestió tancada. En aquesta secció farem una introducció breu a una tecnologia criptogràfica encara més sofisticada que té el potencial d'acabar eliminant alguns dels defectes actuals de Bitcoin.

En la seva encarnació actual, Bitcoin proporciona molt poca anonimitat als usuaris. Tota la informació de les transaccions està disponible públicament. Però, com veurem, això no és un tret necessari, perquè es pot resoldre fent servir la tecnologia de *proves de coneixement nul* (*zero-knowledge proofs*).

5.1 Proves de coneixement nul

De vegades volem demostrar a algú que sabem una cosa sense donar-ne més informació que el fet que la sabem. Un exemple del món físic (per distingir-lo del món dels ordinadors) pot ser la tasca de demostrar a un nen que sabem on és en Wally sense revelar-ne la localització en el pòster. Com ho farem? Si disposéssim d'una fotocopiadora, podríem fer una còpia del pòster, tallar la figura del Wally, destruir la resta de la còpia, i només ensenyar la figura al nen [8]. Perquè sigui un bon protocol, caldria que no donés cap informació sobre la ubicació del Wally al nen, i també convèncer-lo que no hem fet trampa. En el món físic no és fàcil fer-ho de manera precisa. Per exemple, hem de posar condicions sobre el que pot i no pot haver-hi dins l'habitació de la fotocopiadora perquè el nen estigui segur que, posem per cas, no hem retallat la figura del Wally d'un altre pòster.

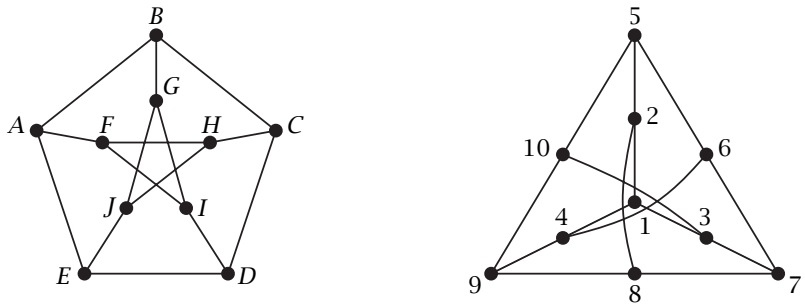

FIGURA 3: Dos grafs que són isomorfs.

El Premi Turing de l'any 2012 (considerat l'equivalent del Premi Nobel en la informàtica) es va atorgar als inventors de la definició matemàtica del concepte de *proves de coneixement nul*. En un article de l'any 1985, Goldwasser, Micali i Rackoff [4] van donar el primer exemple d'aquest tipus de demostració per a dos problemes computacionals de la teoria de números: el problema del residu quadràtic i el problema del no-residu quadràtic. Donat un número a i una base N, es vol demostrar que existeix (pel problema del residu quadràtic) o que no existeix (pel problema del no-residu quadràtic) un número b tal que $b^2 \equiv a \pmod{N}$.

La definició de *prova de coneixement nul* fa servir el concepte de *sistema de demostracions interactives de la complexitat computacional*. Per no haver d'entrar en aquests temes, aquí en farem una descripció informal mitjançant un exemple. L'any 1986 Goldreich, Micali i Wigderson [3] van presentar les primeres proves de coneixement nul per a problemes que no venien de la teoria de números. En concret, ho van fer per als problemes d'isomorfisme i no-isomorfisme de grafs.[6]

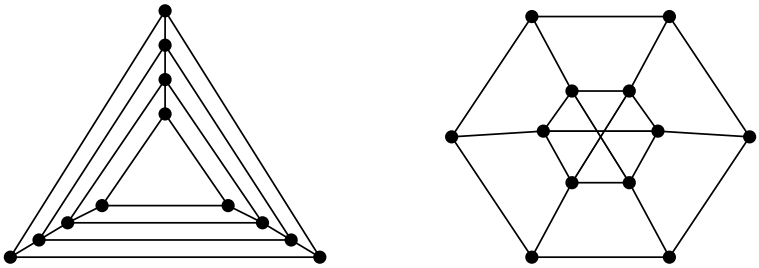

FIGURA 4: Dos grafs que no són isomorfs.

Diem que dos grafs són isomorfs si existeix una correspondència (bijecció) entre els vèrtexs dels dos grafs tal que cada parell de vèrtexs connectats en un dels grafs correspon a un parell de vèrtexs connectats en l'altre i cada parell de vèrtexs no connectats en l'un correspon a un parell de vèrtexs no connectats en l'altre. D'aquest tipus de correspondència se'n diu *isomorfisme*. També podem pensar que els dos grafs realment són representacions diferents del mateix graf, amb diferents etiquetes als vèrtexs. Per exemple, a la figura 3, una correspondència entre els dos grafs que és un isomorfisme és: A amb 1, B amb 2, C amb 5, D amb 6, E amb 4, F amb 3, G amb 8, H amb 10, I amb 7, J amb 9. D'altra banda, els dos grafs de la figura 4 no són isomorfs. Una manera de convèncer-nos d'aquest fet és notar que un dels grafs conté triangles i l'altre no.

Per a grafs petits, sempre podem comprovar si dos grafs són isomorfs o no mirant totes les possibles correspondències entre els vèrtexs. Per a dos grafs de n vèrtexs hi ha $n!$ possibles correspondències i, per tant, aquesta estratègia no és pràctica per a n gran. Hi ha estratègies que funcionen per a molts parells de grafs, però no es coneix cap algoritme eficient (que trigui temps polinòmic en la mida dels grafs) que pugui detectar tots els parells de grafs isomorfs.

Suposem que tenim dos jugadors: la demostradora (*prover*) Peggy i el verificador (*verifier*) Vic. La Peggy coneix un isomorfisme entre els dos grafs G_1 i G_2 i vol demostrar al Vic que els grafs són isomorfs sense donar-li cap informació d'aquest isomorfisme, tret del fet que existeix. El protocol és el següent.

6 A més, van provar que aquestes demostracions existeixen per a tots els llenguatges NP, començant pel problema de coloració de grafs.

La Peggy envia al Vic un graf H que és isomorf a G_1 i a G_2. Perquè no doni cap informació al Vic sobre l'isomorfisme entre G_1 i G_2, la Peggy pot generar-lo etiquetant de manera aleatòria els vèrtexs de G_1. Com que ella sap com ha generat H, ella té un isomorfisme entre els vèrtexs de G_1 i de H. Com que també coneix un isomorfisme entre els vèrtexs de G_1 i G_2, també pot trobar un isomorfisme entre els vèrtexs de H i G_2.

En el segon pas, el Vic demana a la Peggy que li ensenyi l'isomorfisme entre H i G_1 o entre H i G_2. Ell escull quin dels dos li demana.

En cap cas la Peggy li envia tots dos isomorfismes perquè, amb tots dos, el Vic podria recuperar l'isomorfisme entre G_1 i G_2. Però el que és clar és que, si els dos grafs són realment isomorfs, la Peggy sempre pot enviar l'isomorfisme que li hagin demanat.

Si la Peggy intenta convèncer el Vic que dos grafs són isomorfs quan realment no ho són, el Vic l'enxamparà almenys en el 50 % dels casos independentment del que faci la Peggy. Per estar encara més segur que la Peggy no menteix, el Vic pot repetir els dos passos tantes vegades com vulgui. Si en k repeticions amb diferents H la Peggy sempre aconsegueix enviar-li un isomorfisme, el Vic estarà $100 \times \left(1 - \frac{1}{2^k}\right)$ % segur que la Peggy no menteix (perquè la probabilitat que ella endevini totes les preguntes del Vic és 1 entre 2^k). Per exemple, amb $k = 10$ repeticions, el Vic té una probabilitat del 99,9 % d'enxampar la Peggy.

Aquest protocol té les propietats següents que fan que sigui una prova de coneixement nul:

- Si G_1 i G_2 són isomorfs, el Vic sempre estarà convençut d'aquest fet al final.
- Si G_1 i G_2 no són isomorfs, el Vic estarà convençut que ho són només amb una probabilitat mínima (tan petita com vulgui).
- La comunicació del protocol tal com la veuria un tercer la pot generar el Vic tot sol i, per tant, el Vic no obté cap informació que no tingui ja.

La propietat de coneixement nul és l'última en aquesta llista i és la més difícil de demostrar en la majoria dels casos. S'ha de demostrar que hi ha una simulació eficient de la comunicació. És a dir, es pot generar una seqüència d'intercanvis amb la mateixa distribució de probabilitat que la del protocol real, i a més això es pot fer tenint no més informació que la que té el Vic. En el cas especial de l'isomorfisme de grafs això resulta fàcil. El Vic pot simular tota la comunicació generant primer les seves preguntes i després els grafs H isomorfs al graf que correspon a cada pregunta, junt amb l'isomorfisme entre H i aquest graf.

5.2 Proves de coneixement nul per a l'anonimitat de les criptomonedes

Una de les criptomonedes més sofisticades, la Zerocash, es va presentar la primavera de 2014 i la seva descripció està disponible a [1]. La idea al darrere de Zerocash és fer servir proves de coneixement nul per demostrar que es té

una moneda sense revelar quina és. D'aquesta manera les transaccions no es poden associar les unes amb les altres i no es poden seguir els traços de les monedes. S'aconsegueix l'anonimitat que li falta al protocol de Bitcoin. Les proves de coneixement nul són bastant més complicades que les que hem vist aquí, per diverses raons. Primer, no són interactives. Segon, estan dissenyades per ser molt més curtes perquè s'han de guardar en la cadena de blocs. I tercer, les afirmacions que s'han de demostrar són bastant més complicades perquè comporten demostrar que un programa s'ha executat correctament.

El Zerocash és un exemple de moneda que encara no s'ha implementat, però demostra les possibilitats de sofisticació que es poden arribar a incorporar a la idea original de Bitcoin si explotem tot el nostre coneixement matemàtic modern.

6 Rellevància

Bitcoin és un dels molts canvis econòmics i socials que es produeixen gràcies a les noves tecnologies. Al segle XXI veiem les llavors d'una societat més comunicativa i més funcional. Alguns exemples són: les inversions de la ciutadania en projectes d'interès comú mitjançant el micromecenatge (*crowdfunding*) en plataformes com Kickstarter; el periodisme obert mitjançant blogs i Twitter; les estructures d'organització de la informació basades en reputació (*reputation systems*) en plataformes com Stack Overflow i Quora. Una altra és la comunitat econòmica que s'està formant al voltant de Bitcoin.

Les possibilitats de canvi que ens garanteix la connectivitat immediata proporcionada per Internet són infinites, però és un terreny desconegut que, a més d'oportunitats, comporta perills. Un d'aquests perills és que no tothom pugui participar en la formulació de les estructures noves i que sigui només una elit tecnològica o financera la que faci servir aquestes oportunitats per crear un món paral·lel amb una democràcia selectiva. És un repte per als joves i per als educadors garantir que la població pugui entendre i actuar envers aquests canvis, que, a més, es produeixen a un ritme cada cop més vertiginós.

Referències

[1] Ben-Sasson. E.; Chiesa, A.; Garman, C.; Green, M.; Miers, I.; Tromer, E.; Virza, M. «Zerocash: Decentralized Anonymous Payments from Bitcoin». A: *Proceedings of the 2014 IEEE Symposium on Security and Privacy*. Washington, DC: IEEE Computer Society, 2014, 459–474.

[2] Dwork, C.; Naor, M. «Pricing via Processing or Combatting Junk Mail». A: Brickell, E. F. (ed.). *Advances in Cryptology - CRYPTO' 92* (12th Annual International Cryptology Conference Santa Barbara, California, USA, August 16–20, 1992 Proceedings). Berlín: Springer, 1983, 139–147. (Lecture Notes in Comput. Sci.; 740)

[3] Goldreich, O.; Micali, S.; Wigderson, A. «Proofs that yield nothing but their validity and a methodology of cryptographic protocol design». A: *Proceedings of the 27th annual Symposium on Foundations of Computer Science (FOCS)*. Washington, DC: IEEE Computer Society, 1986, 174–187.

[4] Goldwasser, S.; Micali, S.; Rackoff, C. «The knowledge complexity of interactive proof systems». A: *Proceedings of the 17th annual ACM symposium on Theory of Computing* (Conference STOC'85, Providence, RI, USA, May 06–08, 1985). Nova York: ACM, 1985, 291–304.

[5] Koblitz, N. «Elliptic curve cryptosystems». *Math. Comp.*, 48 (177) (1987), 203–209.

[6] Miller, V. S. «Use of elliptic curves in cryptography». A: *Advances in Cryptology - CRYPTO' 85 Proceedings*. Berlín: Springer, 1986, 417–426. (Lecture Notes in Comput. Sci.; 218)

[7] Nakamoto, S. «Bitcoin: A peer-to-peer electronic cash system». Report, `bitcoin.org` (2008).

[8] Naor, M.; Naor, Y.; Reingold, O. «Applied kid cryptography or how to convince your children you are not cheating». *Journal of Craptology*, 0 (1) (1999).

[9] Rivest, R. L.; Shamir, A.; Adleman, L. «A method for obtaining digital signatures and public-key cryptosystems». *Comm. ACM*, 21 (2) (1978), 120–126.

[10] Ron, D.; Shamir, A. «Quantitative Analysis of the Full Bitcoin Transaction Graph». A: Sadeghi, A.-R. (ed.). *Financial Cryptography and Data Security* (17th International Conference, FC 2013, Okinawa, Japan, April 1–5, 2013, Revised Selected Papers). Berlín: Springer, 2013, 6–24. (Lecture Notes in Comput. Sci.; 7859)

[11] Shor, P. W. «Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer». *SIAM J. Comput.*, 26 (5) (1997), 1484–1509.

[12] Silverman, J. H. *The arithmetic of elliptic curves*. 2a ed. Dordrecht: Springer, 2009. (Graduate Texts in Mathematics; 106)

Departament de Matemàtica Aplicada i Anàlisi
Universitat de Barcelona
Gran Via de Les Corts Catalanes, 585
elitza.maneva@gmail.com