Bitcoin & Crittografia

1. Monete digitali

1.1 Centralizzazione e decentralizzazione - La prima proposta di "moneta digitale" è del 1983 [1]: sfruttando una proprietà matematica di RSA, D. Chaum inventa la "firma cieca" (*blind signature*) e ne evidenzia l'uso per pagare beni/servizi con documenti digitali autenticati da un'autorità fidata ed atti a garantire:

- la non tracciabilità dei pagamenti,
- la certezza del cambio con moneta reale
- l'impossibilità di fare più di un acquisto usando copie di una stessa moneta (il cosiddetto attacco con doppia spesa).

Nei venticinque anni successivi, a seguito dapprima dell'avvento del WWW e poi dell' *e-commerce,* si riscontrano decine di proposte per fare pagamenti sicuri su Internet: molte restano solo sulla carta, per alcune vengono sviluppati prototipi, poche infine entrano in servizio. Caratteristica comune a tutte è la presenza di autorità fidate, poste al di sopra degli utenti ed incaricate in vario modo di garantire la sicurezza del sistema.

Un riferimento importante per lo sviluppo di queste nuove monete è il modello con cui nel 1996 un gruppo di ricercatori del NIST ha proposto di rendere impossibile l'attacco con doppia spesa [2]: un'autorità fidata opera come una "zecca" coniando continuamente monete per sostituire quelle che ritira subito dopo che sono state spese.

E' proprio la presenza di autorità a venire progressivamente identificata come il maggior difetto di questi nuovi sistemi di pagamento: tutti, prima o poi, vengono abbandonati. L'autorità, qualsiasi sia il compito svolto, costituisce infatti un "punto singolare" nell'architettura del sistema ed è quindi causa di inefficienza, di inaffidabilità e di complessità d'uso.

Due casi sono emblematici. Il fallimento nel 1998 della ditta Digicash, fondata da Chaum per commercializzare la sua idea di e-cash, è stato una conseguenza della complessità del sw e del hw da installare presso gli utenti e della necessità di coinvolgere un servizio centralizzato in ogni pagamento. All'inizio degli anni 2000, poco dopo l'entrata in servizio e nonostante i successivi perfezionamenti, Visa e Master Card hanno deciso di dismettere il protocollo SET, che avevano promosso con l'intendimento di creare uno standard de facto per i pagamenti online con carte di credito: cause di questo flop sono state l'affermazione del più semplice protocollo TLS, la complessità delle azioni richieste agli utenti, l'inefficienza ed il costo dei servizi centralizzati.

La svolta decisiva a favore della moneta virtuale si verifica nel 2009, quando un progetto *open source*, seguendo le linee guida indicate in un articolo pubblicato nel 2008 da un autore rimasto anonimo [3], mette a dispozione di tutti un innovativo servizio per trasferire valuta su Internet. L'immediato e ampio successo ottenuto da Bitcoin determina negli anni successivi una crescita esponenziale del numero di nuove *cryptocurrencies.*

Aspetto caratterizzante di tutte le criptovalute è l'adozione del paradigma della "decentralizzazione", in contrapposizione a quello della "centralizzazione" di supporti e servizi adottato in precedenza. Tre sono le proprietà della seconda generazione di monete digitali:

- non viene coinvolta alcuna autorità,
- non ci sono punti di singolarità,
- la sicurezza si basa sull'uso di primitive e protocolli crittografici "forti".

Nel seguito faremo riferimento al solo Bitcoin. Lasciando da parte le tante accuse che gli sono state rivolte (attività illegale, bolla finanziaria, strumento per evadere il fisco e per riciclare danaro sporco, causa di un enorme spreco di energia, ecc.), in questa sede consideriamo ben più interessante evidenziare come l'uso, spesso con modalità innovative, di noti e precedenti algoritmi crittografici abbia dato sicurezza e solidità alla sua implementazione.

Per una visione più ampia si rimanda agli e-book [4],[5]. Per chi vuole provare a svilupparne delle parti sono utili la libreria "bitcoinj"[26] ed i blog [6],[7].

1.2 Bitcoin – La prima cosa da capire bene è che la decentralizzazione ha comportato la totale eliminazione di intermediari nelle transazioni finanziarie (*transactions*). Per sottolineare la portata di questa innovazione, supponiamo che chi vuole trasferire monete virtuali debba predisporre e firmare un documento digitale contenente

- un numero di serie,
- una data,
- il suo nominativo,
- quello del beneficiario,
- l'ammontare del trasferimento.

A prima vista, in totale assenza di intermediari (*middleman*) fidati (una banca/società finanziaria che garantisca la disponibilità del denaro, una autorità di certificazione che garantisca l'identità del firmatario, un servizio di marcatura temporale che garantisca la data del documento, un giudice che risolva dispute sulla sua autenticità), non si capisce proprio come si possa aver fiducia che sia autentico un messaggio di questo tipo inoltrato su Internet. Il grande merito di Bitcoin è l'aver dimostrato che tutte queste garanzie non sono indispensabili: un documento firmato digitalmente può essere considerato valido e rappresentare quindi un reale trasferimento di monete virtuali quando tutti possono verificare che la firma è autentica, che il firmatario ha la disponibilità dell'ammontare trasferito, che è la prima volta che lo spende e che il beneficiario potrà in seguito spenderlo a sua volta.

Per ottenere questo Bitcoin prevede

- che al suo interno vi sia, in chiaro, un "libro mastro" (*ledger*) a prova di manomissione, ove vengono alloggiate via via in ordine temporale soltanto transazioni valide;
- che una transazione, per essere considerata valida, debba necessariamente indicare quale di quelle già presenti nel ledger consente di riconoscere al firmatario la proprietà e la disponibilità delle monete trasferite;
- che un semplice sw di supporto, detto portafoglio (*wallet*), metta chiunque in grado di fare transazioni valide, di firmarle e di trasmetterle al libro mastro tramite una rete P2P (*peer-to-peer network*), sovrapposta ad Internet.

Più avanti esamineremo gli algoritmi ed i protocolli crittografici che rendono inalterabile il libro mastro, che consentono di controllare con sicurezza la validità di una transazione e che generano e proteggono le chiavi di firma degli utenti.

E' infatti più urgente eliminare due possibili dubbi. Come è possibile prevedere un punto di singolarità come un database in una architettura decentralizzata? Come è possibile fare a meno di un'autorità fidata che ne controlli l'ingresso?

La prima risposta è semplice: il libro mastro è un'unica base di dati dal punto di vista logico ed un sistema distribuito dal punto di vista della implementazione. Bitcoin prevede che chiunque possa diventare un "minatore", cioè uno che usa le sue risorse informatiche per alloggiare una copia del libro mastro e per partecipare al protocollo, detto del "consenso", che consente di mantenerla sincronizzata con le altre. La seconda risposta è una conseguenza della prima: ogni minatore decide autonomamente se immettere o meno nella sua copia del ledger le transazioni trasmesse dalla rete.

In Bitcoin esistono dunque regole di corretto impiego [8], utenti interessati a rispettarle e minatori incentivati a farlo da premi e compensi in monete. Possono però esserci anche malintenzionati che per qualche loro motivo decidono di violarle.

Il sistema è tollerante alle manomissioni (*tamper resistant*) fino a quando i malintenzionati sono una minoranza. Per rispettare il paradigma della decentralizzazione, tutte le azioni si basano infatti su decisioni prese dalla maggioranza:

- la sincronizzazione dei DB assume come corretto il contenuto del maggior numero di copie;
- una transazione è corretta solo se la ritene tale la maggioranza degli utenti;
- il libro mastro è inalterabile fino a quando non viene manomesso più del 50% delle copie.

La totale assenza di “punti singolari di rottura” attribuisce a Bitcoin anche la proprietà di tolleranza ai guasti (*fault tolerance*) .

1.3 Primitive crittografiche – Tre sono le proprietà che rendono Bitcoin un “sistema sicuro”:

1. l'inalterabilità del libro mastro
2. la controllabilità delle transazioni
3. la sicurezza delle chiavi

Nel seguito le esamineremo separatamente. E' però utile anticipare subito che tutta la sicurezza si basa fondamentalmente su due soli algoritmi crittografici: l'hash e la firma.

HASH - Per il calcolo di impronte sono usati SHA-2 (con 256 e 512 bit in uscita) e RIPEMD (con 160 bit in uscita), algoritmi per i quali a tutt'oggi sono considerati intrattabili i seguenti problemi:

- trovare due documenti diversi con eguale hash,
- data una stringa , trovare un documento di cui sia l'hash,
- dato un documento, trovarne uno diverso, ma con lo stesso hash.

Sei sono le applicazioni usate in Bitcoin:

- la doppia compressione con SHA256 è utilizzata per generare i “puntatori con hash” (v. 2.1), per risolvere il “puzzle computazionale”(v. 2.3), per proteggere con checksum l'indicazione del beneficiario di una transazione (v. 3.1), per comprimere a 256 bit l'input dell'algoritmo di firma ECDSA ;
- la cascata di SHA256 e di RIPEMD160 serve per calcolare gli address (v. 3.1);
- SHA512 viene usato nella generazione deterministica delle chiavi (v. 4.2).

FIRMA - Per autenticare le transazioni Bitcoin impiega l'algoritmo ECDSA (*Elliptic Curve Digital Signature Algorithm*), una variante del precedente standard DSS individuata da NSA nel 1995 [9], [10],[11],[25]. L'adozione in Bitcoin di questo algoritmo (e non di RSA o di DSS) è discesa dal fatto che, a parità di livello di sicurezza, aveva le realizzazioni più efficienti e le chiavi più corte.

In Bitcoin viene usata la curva secp256k1, espressa dall'equazione

$y^2 = x^3 + 7 \bmod p$, con p numero primo di valore prossimo a 2^{256}.

L'insieme dei punti della curva forma un “gruppo ciclico” di ordine $n < p$, su cui è definita un'operazione di somma additiva, commutativa e dotata di elemento neutro. Ogni punto della curva può essere ottenuto moltiplicando per uno scalare un “generatore” G:

$P_x = G+G+....+G = x*G$ (indicato anche con G^x) con $1<x<n$

Per rendere efficiente il calcolo si usano ripetutamente il “raddoppio” e la “somma con G”.

ESEMPIO: 71*G= G+2*(35*G)= G+2*(G+2*(17*G))= G+2*(G+2*(G+2*(8*G))) ...

ECDSA è formato da tre algoritmi:

1. *ECDSAgenkeys* ha in ingresso un numero di 256 bit prodotto da un CSPRNG (*Cryptographically Secure Pseudo-Random Number Generator*) e lo riproduce in uscita come chiave privata S; la chiave pubblica P, ottenuta eseguendo il prodotto S*G, è resa disponibile su una seconda uscita tramite le sue coordinate di 32 byte ciascuna. La indeducibilità di S da P è garantita dalla difficoltà di risolvere il problema del logaritmo discreto: il livello di sicurezza è di 128 bit.
2. *ECDSAsign* ha in ingresso l'hash con SHA256 del messaggio da firmare, la chiave privata ed un numero a caso k di 32 byte che deve essere usato una volta sola. La firma fornita in uscita è la coppia di due dati (r e s) di 256 bit ciascuno.
3. *ECDSAverify* giudica vera o falsa una firma elaborando la coppia <r,s>, la chiave pubblica P e l'impronta del messaggio.

ECDSA è accrediato di un livello di sicurezza di 128 bit. Non c'è una dimostrazione teorica. Una garanzia indiretta che l'algoritmo è sicuro discende dal suo grandissimo uso in Bitcoin ed in Ethereum: non sarebbe infatti passato sotto silenzio se qualcuno fosse riuscito ad incassare le monete di un altro! Non fa testo la violazione delle chiavi delle *playstation* Sony [26]: nelle due firme era stato erroneamente usato lo stesso valore per il numero a caso k.

2. Inalterabilità del libro mastro

2.1 Hash pointer – Con "puntatore con impronta"(*hash pointer*) si intende una struttura di dati avente il duplice compito di indicare dov'è stato alloggiato un documento in un database e qual'era la sua impronta al momento della registrazione.

La proposta di questo particolare meccanismo crittografico è stata fatta nel 1991 da Haber e Stornetta [12] per evidenziare come sia possibile realizzare un servizio sicuro di marcatura temporale di documenti senza far ricorso alla firma digitale.

L'idea di base è stata quella di mettere in testa ad ogni nuovo documento da registrare il puntatore con impronta dell'ultimo registrato.

Con questo accorgimento si riesce a creare una "lista" di documenti ordinati cronologicamente, scorrendo la quale è possibile rilevare la presenza di intrusioni (*tamper detection*): se un intruso modifica anche un solo bit di uno qualsiasi dei record, si modifica infatti anche la sua impronta e salta di conseguenza la corrispondenza con l'hash pointer alloggiato nel record registrato subito dopo di lui.

Una seconda indicazione è stata quella di rendere di dominio pubblico, e quindi immodificabile, l'hash pointer posto all'ingresso della lista: ciò consente infatti di vanificare il tentativo di mascherare l'intrusione modificando anche tutti gli hash pointer compresi tra il record alterato e l'ingresso della lista.

Per accedere più velocemente ad ogni documento alloggiato nel DB è stato infine proposto di inserirvi via via "blocchi" contenenti più documenti disposti nella struttura ad albero brevettata da Merkle nel 1979 [13].

Bitcoin recepisce in toto questi suggerimenti: nel ledger vengono introdotti e collegati con hash pointer blocchi contenenti diverse transazioni.

Gli hash pointer, calcolati con SHA256, sono disposti ad albero binario: ogni nodo "foglia" è l'hash pointer di una transazione ed ogni nodo "ramo" è l'hash dei 64 byte formati dalla concatenazione dei suoi due nodi "figli"; esiste di conseguenza un nodo "radice", di 32 byte, che costituisce l'impronta dell'intero blocco.

Dato che SHA256 si comporta come un "oracolo casuale", tale impronta può essere considerata una caratteristica "univoca" di ogni blocco: per il paradosso del "giorno di compleanno" la probabilità di avere due blocchi con la stessa impronta è 2^{-128}, un numero minore di 1 che in decimale ha 38 zeri dopo la virgola!

L'hash pointer radice è inserito nella intestazione del blocco (*block heade*r), unitamente al puntatore con l'impronta del blocco introdotto prima di lui e ad altri tre dati (il *timestamp*, il *target* ed il *nonce*) di cui parleremo nel prossimo paragrafo.

Ciò rende il libro mastro una catena di blocchi (*blockchain*) progressivamente allungata dal continuo arrivo di nuovi blocchi; a marzo del 2018 la capacità della memoria ove alloggiarla aveva raggiunto i 180 milioni di byte.

La prima transazione di ogni blocco, detta "*coin base*", ha come beneficiario il minatore che ha costruito il blocco e l'ammontare è il compenso che gli viene riconosciuto, a patto però che la maggioranza degli altri minatori giudichi corretto il lavoro che ha svolto.

Particolarmente importante dal punto di vista della sicurezza è l'innovazione che Bitcoin ha introdotto nel modello di Haber e Stornetta: l'impossibilità di modificare gli hash pointer è ottenuta obbligando l'intruso a fare un calcolo complesso, un puzzle difficile da risolvere e facile da verificare, esattamente lo stesso che sono obbligati a fare i minatori per mantenere sincronizzate le loro copie di libro mastro (v. 2.3).

Questo accorgimento ha origini lontane. La prima proposta di aumentare la sicurezza di un servizio facendo fare calcoli onerosi a chi vuole accedervi è del 1992: l'obiettivo era quello di difendere dallo spam il servizio di posta elettronica [14].

La risoluzione di un puzzle è stata proposta per la prima volta nel 1998 da Wei Dai [15] e ripresa poi nel 2005 da N. Szabo [16].

2.2 La sincronizzazione – Per il paradigma della decentralizzazione non può esserci un'autorità che mantiene sincronizzate le migliaia e migliaia di copie del libro mastro sparse in tutto il mondo. Si è già detto che l'incarico è direttamente affidato ai minatori che le hanno in consegna. Vediamo ora, con un certo dettaglio, quali sono le attività che ogni minatore deve svolgere ripetutamente:

1. controllare le transazioni diffuse sulla rete e memorizzare in un buffer solo quelle che giudica corrette;
2. inserire in un blocco un certo numero delle transazioni presenti nel buffer;
3. provare a risolvere un puzzle che in media richiede 10 minuti di calcoli.
4. trasmettere il blocco che ha preparato se trova la soluzione e se nel frattempo non gli è arrivato un blocco trasmesso da un altro minatore; interrompere i calcoli se si verifica questo evento;
5. controllare il blocco ricevuto e, se è ok, aggiungerlo in testa alla blockchain, memorizzandone l'hash per poterlo poi inserire come puntatore nel blocco successivo.

Due osservazioni.

Per rispettare fino in fondo il paradigma della decentralizzazione, il minatore che propone un nuovo blocco dovrebbe essere scelto a caso: a tal fine dunque tutti dovrebbero avere la stessa probabilità di vincere la gara computazionale, cosa che vedremo non essere più tanto vera.

L'imporre che i calcoli durino in media 10 minuti è il risultato di un compromesso: da un lato si riesce a garantire che l'immissione di un nuovo blocco nella blockchain si verifichi quando il sistema ha già raggiunto una condizioni di stabilità, da un altro lato si rende possibile misurare con buona precisione quando è stata fatta ogni transazione.

2.3 Il puzzle computazionale - Il puzzle si riferisce a due dati, il "target" ed il "nonce", contenuti nell'intestazione di ogni blocco:

- il target è un numero di valore $2^{(256-n)}$-1, cioè un numero di 256 bit formato da n "zeri" seguiti da (256 – n) "uni";
- il nonce è un dato il cui valore alla fine dei calcoli deve rendere minore o uguale al target l'hash con SHA256 della concatenazione del Merkle root, del hash dell'intestazione del blocco precedente, del timestamp, del target e del nonce.

Il target viene ricalcolato ogni 2016 nuovi blocchi inseriti nel libro mastro: se il tempo trascorso (calcolato come differenza dei time stamp del primo e dell'ultimo blocco) è superiore a 2 settimane (10x6x24x14 minuti = 2 settimane) il target viene innalzato, se è inferiore viene diminuito.

Per le proprietà di sicurezza di SHA256 il puzzle può essere risolto solo ricorrendo alla forza bruta ed avendo molta pazienza. Vediamo perché.

La probabilità di ottenere, con un solo tentativo, l'hash desiderato è data dal rapporto tra i casi favorevoli ed i casi possibili: $p = 2^{-n}$.

La ripetizione dei tentativi in caso di insuccesso può essere modellata con un processo di Bernoulli: la probabilità di risolvere il problema dopo k-1 insuccessi è $P(N=k) = p.(1-p)^{k-1}$.

Il valore atteso per il n° di tentativi da fare prima di ottenere l'hash desiderato è dunque 2^n.

La probabilità che un certo minatore vinca la gara è infine data dal rapporto tra la potenza di calcolo della sua macchina e quella di tutti gli altri minetori messi insieme.

Per capire quanto sia complesso risolvere il puzzle, si pensi che nel 2009 il target era 2^{224}-1 ed il numero medio di impronte da calcolare era $2^{32} = 4{,}3.10^9$; impiegando un desktop a tempo pieno occorrevano infatti mediamente i 10 minuti richiesti dalle specifiche del sistema.

Il miraggio di impossessarsi del premio ha spinto successivamente i minatori ad usare calcolatori e dispositivi periferici sempre più potenti, determinandosi così una continua crescita della frequenza di calcolo degli hash ed una conseguente diminuzione del target. Oggi è richiesto che l'hash abbia almeno 72 zeri in testa: occorre infatti prendere atto che alcuni minatori impiegano sistemi paralleli contenenti migliaia e migliaia di SHA256 realizzati in hw da specifici circuiti

integrati (*Application Specific Integrated Circuit*). Queste macchine riescono a calcolare 10^{19} hash in dieci minuti; una odierna "normale" workstation impiegherebbe anni per risolvere il puzzle.

Tutto questo ha però creato un problema di sicurezza: l'enorme crescita degli investimenti e delle successive spese per l'alimentazione ed il raffreddamento di questi dispositivi elettronici ha fatto scendere in campo un numero relativamente basso di vere e proprie imprese industriali e messo quindi in pericolo il principio della decentralizzazione: l'aumento della probabilità di una loro vittoria rende infatti dubbia l'ipotesi che a decidere quale nuovo blocco deve entrare nella blockchain sia un minatore scelto a caso. E' però giusto anche riconoscere che l'enorme crescita della complessità dei calcoli e dei costi di gestione ha reso sempre più difficile e meno vantaggioso il portare attacchi al sistema.

Un notevole aiuto alla decentralizzazione (e quindi alla sicurezza) è stato per fortuna dato dai numerosi *mining pool*, all'interno dei quali migliaia di singoli minatori, opportunamente coordinati, condividono parte della loro potenza di calcolo per aumentare la velocità di ricerca della soluzione di uno stesso puzzle.

Per avere la prova (*proof of work*) che il proponente di un nuovo blocco ha risolto il puzzle, i minatori, dopo aver verificato la correttezza di tutte le transazioni, dell'impronta del blocco e del target, devono calcolare l'hash dell'intestazione e confrontare il risultato con il target.

Non sono possibili inganni sulla prova che il lavoro di miniera è stato svolto correttamente: a seguito dei controlli precedentemente indicati, vengono infatti immediatamente scoperti i tentativi

- di falsificare transazioni (v. 3.3),
- di copiare il nonce per risparmiarsi i calcoli,
- di precalcolare il nonce per avvantaggiarsi nella gara con gli altri minatori.

Se la verifica di un blocco fallisce, i minatori cestinano il blocco ricevuto e riprendo i loro calcoli. Se tutto va bene controllano se l'hash pointer contenuto nell'intestazione del nuovo blocco è proprio l'impronta dell'ultimo blocco che hanno inserito nel ledger. In caso affermativo esprimono un consenso implicito appendendo il blocco nella loro copia del libro mastro e cancellando dal loro buffer tutte le transazioni che contiene. In caso contrario dichiarano il blocco "orfano" e lo parcheggiano temporaneamente in un buffer.

2.4 Tamperproof - Si è già detto che per rendere non rilevabile una modifica apportata ad un qualsiasi blocco della blockchain l'intruso deve risalire la catena fino all'ingresso modificando gli hash pointer posti in testa ai blocchi che via via incontra. Possiamo ora precisare che per ogni blocco che incontra l'intruso deve anche risolvere il relativo puzzle: il nuovo valore dell'hash pointer modifica infatti a sua volta l'impronta dell'intestazione di cui fa parte ed è altamente probabile che quest'ultima non risulti più minore o uguale al target.

Si crea così una gara di velocità tra l'attaccante che vuole risalire fino all'ingresso apportando modifiche ed i minatori che stanno progressivamente allungando la blockchain. Se la potenza di calcolo complessiva dei minatori onesti è molto superiore a quella dell'intruso (cosa che deve essere vera per ipotesi e che fino ad oggi si è dimostrata tale), l'aggiunta di un nuovo blocco alla blockchain si verificherà prima che l'intruso riesca a risolvere un puzzle.

I blocchi che l'intruso è riuscito ad alterare resteranno dunque in conflitto con quelli esistenti e gli altri minatori potranno autonomamente rigettarli. Tutto questo consente di riconoscere alla blockchain la proprietà di essere "a prova di intrusione"(*tamperproof*).

Nel suo pioneristico articolo Satoshi Nakamoto, facendo riferimento al classico problema statistico della "rovina del giocatore" [17], ha evidenziato che la probabilità dell'intruso di arrivare fino all'ingresso della blockchain diminuisce secondo una progressione geometrica all'aumentare del numero di blocchi che deve modificare. Una regola pratica dedotta da questo modello suggerisce a chi ha venduto qualcosa di consegnarla al compratore solo dopo aver visto nella blockchain che il blocco contenente il suo pagamento ha avuto almeno 6 "conferme", cioè è stato seguito dalla registrazione di almeno 6 nuovi blocchi.

2.5 Biforcazioni e attacco con doppia spesa - Di norma ogni blocco ha un unico successore.

Ogni tanto però può capitare che due minatori risolvano quasi contemporaneamente il puzzle e diffondano sulla rete due blocchi diversi, ma con lo stesso hash pointer al blocco in quel momento in testa al libro mastro.

Per la latenza della rete l'ordine di arrivo dei due blocchi nei vari nodi sarà casuale: il primo arrivato, se corretto, verrà subito appeso come ultimo anello della catena, il secondo darà inizio ad un ramo secondario della blockchain. Dopo la creazione di una biforcazione (*fork*) è molto probabile che arrivi un blocco solo: uno dei due rami diventerà così il più lungo ed un algoritmo distribuito obbligherà i minatori a considerarlo come "principale" e a non prendere più in considerazione il blocco contenuto nell'altro ramo. Se per caso si ripetesse l'arrivo contemporaneo di due blocchi non succede niente di male: il riallineamento delle diverse copie della blockchain verrà solo ritardato.

Un malintenzionato potrebbe pensare di sfruttare questo comportamento del sistema per portare un attacco con doppia spesa: dopo aver fatto un primo e corretto trasferimento di monete ad un suo creditore, l'attaccante subito dopo, con un secondo ed illecito trasferimento, indica se stesso o un complice come beneficiario delle stesse monete e, corrompendo uno o più minatori disonesti, riesce a creare una situazione di biforcazione in cui la sua seconda transazione si venga a trovare nel ramo più lungo e la prima in quello più corto.

L'attacco gli riesce però solo a metà: quale delle due transazioni si troverà nel ramo più lungo sarà infatti deciso non da lui, ma dall'ordine con cui sono state ricevute dalla maggioranza dei minatori onesti.

3 – Il controllo delle transazioni

3.1 Address – Chi vuole trasferire e ricevere BTC deve

- installare nel suo PC un apposito sw per interfacciarsi con la reteP2P,
- generare una coppia di chiavi ECDSA con cui firmare e far verificare le sue transazioni,
- generare e comunicare a chi gli deve trasferire monete un dato di 20 byte, detto "indirizzo" (*address*), da usare come indicazione del beneficiario.

Nella transazione standard P2PKH (*Pay To Public Key Hash*), a cui nel seguito faremo riferimento, gli utenti generano questo pseudonimo comprimendo due volte la chiave pubblica P:

address = RIPEMD160(SHA256(P)).

In Bitcoin gli utenti sono anonimi: non esistono *account* e non c'è quindi bisogno di un'autorità di registrazione. Non servono i certificati e le autorità di certificazione. Non c'è infine bisogno di giudici, perchè le transazioni, una volta firmate, non possono essere ripudiate.

Un utente è dunque identificato dal solo fatto che usa una certa coppia di chiavi. Può venire un dubbio: dato che la chiave privata S è un numero scelto a caso e che non c'è alcun controllo centralizzato, due utenti potrebbero trovarsi ad averla uguale ed uno dei due potrebbe così riuscire a spendere anche i soldi dell'altro. E' vero, ma in pratica può essere considerato impossibile: la probabilità che ciò capiti è infatti 2^{-128}.

L'integrità dell'address è una esigenza assoluta: se infatti si modifica anche un solo bit (nella comunicazione, o nella successiva memorizzazione) le monete che il beneficiario sta aspettando sono irrimediabilmente perse!

La prima difesa è la rilevazione di errori: ad ogni address vengono affiancati, come *check-sum*, i primi 4 byte della sua impronta calcolata con SHA256.

La seconda difesa è la prevenzione: nella comunicazione scritta dell'address, per evitare fraintendimenti, la stringa di bit così ottenuta viene convertita in "base 58", cioè in una stringa di testo formata con 58 degli usuali simboli della "base 64": non sono usati i quattro simboli confondibili (la lettera maiuscola "O" ed il numero "0", la lettera minuscola "l" e la "I" maiuscola) ed i due vietati nelle mail (il + e la barra /).

Spesso la rappresentazione in base 58 è fornita anche con il codice a barre bidimensionale (*QR Code*) per essere acquisita con sicurezza tramite una telecamera; sono naturalmente disponibili convertitori che ripristino la stringa binaria da queste due ultime rappresentazioni.

3.2 Monete in circolazione – Il flusso di BTC da un proprietario ad un altro può essere modellato interconnettendo blocchi dotati di N ingressi e di M uscite. Ogni blocco ha in testa un identificativo e rappresenta una transazione memorizzata nel libro mastro.

Ad ogni uscita di ogni blocco sono associati un numero d'ordine, un quantitativo di monete espresso in "satoshi"(pari a 10^{-8} BTC) e l'address del beneficiario della transazione.

Ad ogni ingresso sono associati un numero d'ordine, un puntatore all'uscita di una precedente transazione in cui compare come beneficiario chi ne ordina il trasferimento con l'attuale transazione, la sua firma e la chiave pubblica per verificarla.

Per ogni blocco esistono due vincoli.

Il primo è che ogni ingresso deve ricevere tutte le monete associate all'uscita del blocco a cui è collegato. Un blocco quindi, con un comportamento simile a quello della zecca proposta in [2], "inghiotte" tutte le monete ricevute in ingresso e provvede a "riconiarle" come monete spendibili in uscita. La somma delle monete in uscita può però essere inferiore a quella delle monete in ingresso: la differenza, quando esiste, apparirà in uscita alla transazione detta "coinbase", costituendo un compenso (*transaction fee*) per il minatore che inserirà la transazione in un blocco della blockchain.

Il secondo vincolo riguarda la proprietà delle monete: in una transazione corretta l'ammontare di denaro trasferito deve coincidere con quello ricevuto dal firmatario in una precedente transazione.

E' dunque la firma di una transazione l'evento che consente di spendere, o di riscattare (*to redeem*), le monete indicate in una precedente transazione. Ovviamente in ogni momento esistono uscite non ancora connesse e quindi quantitativi di monete non ancora spese. Il loro insieme costituisce le monete "in circolazione" in quel momento ed è memorizzato in un file che i minatori aggiornano ogniqualvolta aggiungono un nuovo blocco alla catena.

Questo file consente di fronteggiare un tentativo di "doppia spesa" meno sofisticato di quello discusso in 2.5: l'attaccante spedisce una dopo l'altra le due transazioni e spera che il sistema prenda in considerazione solo la seconda. In realtà anche in questo caso verrà presa in considerazione solo quella che è arrivata per prima nel nodo che diffonderà in rete il nuovo blocco.

3.3 P2PKH: input e output script – Per consentire ai minatori di controllare l'autenticità della firma e la disponibilità delle monete trasferite, tutte le transazioni contengono due semplici programmi, uno associato ad ogni ingresso e detto "input script" (o *scriptSig*) ed uno associato ad ogni uscita e detto "output script" (o *scriptPubkey*) [4], [6], [18]. Le istruzioni del linguaggio di programmazione, detto "Script", sono codificate da un byte.

Consideriamo, per semplicità, una transazione P2PKH con un ingresso ed una uscita. Per controllare se è corretta o meno, i minatori devono sovrapporre l'input script all'output script ed eseguire il codice così ottenuto condividendo l'uso di uno stack. L'input script è formato da due istruzioni (i simboli < > tra cui è racchiuso un dato sottintendono il comando PUSH):

<sign> *inserisce nello stack la firma della attuale transazione*
<P> *inserisce la chiave pubblica che la verifica.* .

L'output script è formato da cinque istruzioni. Le prime 4, eseguite in sequenza, sono

OP_DUP *duplica il top della pila*
OP_HASH *sostituisce il top con il risultato di* RIPEMD160(SHA256(top))
<address> *inserisce nella pila l'address della transazione precedente*
OP_EQUALVERIFY *estrae e confronta i due dati più in alto nello stack.*
Se i dati sono diversi, le monete non sono del firmatario e la transazione viene rigettata.

Se sono uguali l'output script prevede l'esecuzione di una quinta istruzione:

OP_CHECKSIG *verifica la firma <sign> con la chiave P.*

Se la risposta è NO, il documento non è integro e viene rifiutato.

Si noti che una transazione viene cestinata anche se è stato un disturbo, e non un malintenzionato, a renderne incompatibili il testo e la firma. La cosa può realmente accadere, ma non preoccupa minimamente: per un minatore che riceve una copia non integra, ci sono centinaia di altri minatori che ricevono copie integre.

A questo punto è giusto chiedersi se la difesa tramite script può essere aggirata.

Se uno si fa rubare la chiave segreta, la risposta è SI, ma non certo per colpa degli script!

Se invece tutti proteggono bene il contenuto del loro portafoglio (vedremo come nella prossima sezione), la risposta è NO. Il malintenzionato può sicuramente intercettare una transazione che gli arriva dalla rete e alterarne l'address prima di rilanciarla, ma la OP_CHECKSIG le sbarrerà la strada verso il libro mastro; occorrerebbe dunque modificare anche la firma, ma per riuscireci il malintenzionato dovrebbe riuscire a risalire dall'address alla chiave pubblica e dalla chiave pubblica alla chiave segreta, due problemi non risolvibili con il livello di sicurezza di 128 bit.

Eseguendo i due script tutti i minatori onesti sono dunque individualmente in grado di rilevare e rigettare ogni tentativo di furto di monete. Come si è già detto, il sistema è sicuro fino a quando sono in maggioranza: le ricompense per il lavoro di miniera sono un apprezzato incentivo; la pesantezza dei calcoli per portare attacchi, un potente disincentivo.

3.4 La transazione P2SH: address e script – Nel secondo tipo di transazione standard, la cosiddetta *Pay_To_Script_Hash* [19], i 160 bit dell'address sono l'hash di uno script e non di una chiave pubblica. In questo modo chi deve ricevere delle monete è messo in grado di stabilire in anticipo come potranno essere spese.

Ampio uso ha avuto il caso particolare della P2SH "multisig", in cui lo script esprime la seguente regola: "la transazione che spenderà queste monete deve essere firmata da almeno M delle N≥M chiavi private corrispondenti alle chiavi pubbliche P_1, … ,P_N".

Due casi per evidenziarne l'utilità ed il contributo alla sicurezza:

- con la regola "3 su 5" una complessa organizzazione aziendale può far decidere a maggioranza se e come spendere le sue monete;
- con la regola "2 su 3" un utente può proteggersi dalla perdita di una chiave privata predisponendone tre coppie e prevedendo la presenza sulla transazione di almeno due firme,
- con la regola "2 su 2" un utente preoccupato dal fatto che una sua chiave segreta possa essere scoperta, può rendere corrette le sole transazioni in cui ha apposto due firme.

Nella P2SH "multisig" il beneficiario è individuato dall'impronta dello script:

address:= RIPEMD160(SHA256(*script*))

script:= OP_M <P_1>...<P_N> OP_N OP_CHECKMULTISIG

La transazione che in seguito ne spenderà le monete, avrà al solito due script.

L'input script è <sig_a>...<sig_i> <*script*> e quando è in esecuzione, lascia nello stack le M firme e lo script.

L'output script è OP_DUP OP_HASH <*address*> OP_EQUAL e quando è in esecuzione (al solito dopo l'input script),

1. duplica il top,
2. spinge dentro allo stack l'address della transazione puntata,
3. estrae e confronta i due dati più in alto;
4. in caso di risposta "true", l'esecuzione dello script rimasto nello stack controlla se tutte le M firme sono verificabili.

4 – La sicurezza delle chiavi

4.1 Il portafoglio - I BTC di ogni utente sono memorizzati nella blockchain sotto forma di "transazioni con uscite non ancora spese": il portafoglio (*wallet*), nonostante il suo nome, non contiene dunque monete, ma coppie di chiavi e SW per gestirle.

Tra le attività richieste all'utente, quattro costituiscono potenziali punti di vulnerabilità: la generazione e la memorizzazione della chiave privata, il calcolo della chiave pubblica, la firma della transazione. Per garantire un buon livello di sicurezza è opportuno attenersi a tre regole:

1. impiegare una sorgente di rumore con alta entropia per generare la chiave privata;
2. alloggiare la chiave privata in ambiente protetto e predisporre una qualche forma di backup, con la consapevolezza che questo creerà un punto singolare di vulnerabilità;
3. usare un computer connesso a Internet per le sole attività di comunicazione dell'indirizzo, di controllo delle conferme e di trasmissione delle transazioni firmate.

Esiste una quarta regola, questa volta per difendere la *privacy*:

4. una coppia di chiavi deve essere utilizzata una volta sola.

L' uso ripetuto di una stessa coppia consentirebbe infatti di individuare il "giro di affari" del proprietario e di dedurre da questo qualche informazione sulla sua identità; in realtà la regola è anche un accorgimento di sicurezza, consentendo di limitare i danni nel caso di perdita o di furto di una chiave privata.

Sono disponibili oggi differenti tipi di wallet in grado di svolgere una o più delle attività richieste dalla gestione delle chiavi. Quali impiegare e come è il frutto di una scelta operata da ogni utente nell'ambito di un compromesso tra la sicurezza e la comodità d'uso.

Esistono anche applicazioni sicure, come l'utilissimo *daemon* messo gratutamente a disposizione di tutti da Bitaddress.org [20], che generano autonomamente coppie di chiavi e ne forniscono la rappresentazione in Base58check ed in QR code; la loro stampa su carta fornisce una forma di backup.

Due metodi di generazione delle chiavi meritano in questa sede una citazione esplicita, il primo per la sua originalità e per la sua efficienza, il secondo per il merito di riproporre in Bitcoin un risultato classico della Crittografia.

4.2 Il portafoglio gerarchico deterministico (Hierarchical Deterministic Wallet)- Per la matematica delle curve ellittiche vale una proprietà notevole: il prodotto di un punto per uno scalare è distributivo rispetto alla somma. In ECDSA una chiave pubblica P_i corrispondente ad una certa chiave privata (S+i) è espressa,

per definizione, da: $P_i=(S+i)*G \bmod p$

Per la proprietà citata si ha: $P_i=(S*G+i*G) \bmod p$

e quindi anche $P_i= P+ i*G \bmod p$

Si noti che, data una coppia S,P ed un intero arbitrario i, è dunque possibile ottenere una nuova coppia con due calcoli separati: la cosa interessante è che per calcolare la chiave pubblica corrispondente a (S+i) è necessario conoscere soltanto P e i. Il portafoglio gerarchico deterministico (BIP 0032 [21]) sfrutta quanto detto per generare un numero grande quanto si vuole di coppie di chiavi calcolando *offline* quelle private e, *online* ed independemente, quelle pubbliche.

Il punto di partenza è il *root seed* (l'unico dato per cui è indispensabile il backup) che viene compresso con SHA512 per ottenere una coppia di chiavi *master* S_m, P_m ed un dato casuale C_m detto *master chain code*:

- i 256 bit più a sinistra delll'impronta costituiscono la chiave privata S_m,
- i 256 bit più a destra costituiscono il dato casuale C_m,
- la chiave pubblica è $P_m = S_m*G$.

Per generare nuove coppie di chiavi vengono impiegati due algoritmi. Il primo, eseguito su una macchina online, parte da una chiave pubblica P ed un dato casuale C (una coppia di questo tipo

è detta “chiave pubblica estesa”) ed è in grado di dedurre, modificando via via un indice i di 32 bit, 2^{32} nuove coppie P_i, C_i:

$$P_i = P+SHA512(P||C||i)*G \bmod p$$

Parallelamente ed indipendemente il secondo algoritmo, questa volta eseguito in ambiente protetto, consente di dedurre dalla chiave privata S, corrispondente a P, e dallo stesso C, le 2^{32} chiavi private estese corrispondenti a quelle pubbliche calcolate dal wallet online:

$$S_i = S+SHA512(S*G||C||i)*G \bmod p$$

Il procedimento può essere ripetuto su entrambe le macchine quante volte si vuole.

Per questo utilissimo metodo esiste un rischio potenziale: la conoscenza di un chain code e di una chiave privata successiva consente di individuare tutte le altre! Per chi non intende correre questo rischio, il metodo prevede di mantenere in ambito protetto, offline, sia la generazione gerarchica delle chiavi private, sia il calcolo delle corrispondenti chiavi pubbliche (*hardened keys*). Ogni tanto l'utente deve dunque trasferire dalla macchina offline a quella online blocchi di nuove chiavi pubbliche, usando per sicurezza una chiavetta USB.

Interessante è il metodo previsto per il calcolo del “seme”. Una volta generata una stringa di bit casuali, la si suddivide in pacchetti di 11 bit ciascuno, che vengono poi utilizzati per indirizzare una memoria contenente 2048 parole comuni della lingua inglese.

La stringa di parole così ottenuta (tipicamente 12 parole, ma è possibile spingersi fino a 24) può essere imparata a memoria (*brain wallet*) e/o scritta su un foglio di carta da tenere in cassaforte. Per ottenere il seme radice di HDW è in uso il classico metodo di protezione delle password [22]: per rendere molto onerosi gli attacchi con dizionario, la stringa, combinata con una eventuale passphrase (il *salt),* viene compressa 512 volte con SHA512.

4.3 Secret sharing - La scomposizione di un dato segreto in N “quote” (*shares*) che vengono separatamente affidate ad altrettanti partecipanti è uno dei più interessanti metodi crittografici di protezione di un segreto [23].

La ricostruzione del segreto è possibile solo se almeno M partecipanti ($M \leq N$) uniscono le loro quote; la sicurezza è dimostrabile facendo ricorso alla Teoria dell'Informazione (*information-theoretic security*): conoscere meno di M quote è come non conoscerne nessuna!

La matematica è quella dei “campi finiti”. E' dato un grande numero primo p ed un polinomio di grado M-1 con coefficienti tenuti segreti:

$$f(x) = a_0 + a_1x + a_2x^2 + .. + a_{M-1}x^{M-1} \pmod p$$

Solo disponendo di almeno M differenti valutazioni del polinomio

$$f(x1), f(x2), ., f(xM),$$

è possibile individuarne tutti i coefficienti a_i con l'interpolazione di Lagrange[24].

I dati f(xi) sono le quote distribuite ai partecipanti; a_0 è il segreto che si vuole difendere.

A titolo d'esempio, facendo però per comodità riferimento al campo infinito dei numeri reali, consideriamo un piano cartesiano e la retta $y = a_0 + a_1x$:

- la conoscenza delle coordinate di un solo punto non consente di tracciare la retta;
- solo la conoscenza di due o più punti consente di evidenziare l'intersezione della retta con l'asse delle ordinate ed individuare così il segreto a_0.

Data dunque una chiave privata di ECDSA, basta suddividerla in 3 quote per avere la garanzia di poter spendere il proprio danaro anche se una quota è andata distrutta o è stata rubata.

L'uso del secret sharing in Bitcoin può essere interpretato come una forma di “decentralizzazione” di un segreto.

Nella piattaforma GitHub è disponibile una libreria che consente di calcolare le quote di una chiave privata rappresentata in Base58Check.

Bitaddress.org mette a disposizione l'intero servizio.

Riferimenti

1. Le monete virtuali ….....

[1] Chaum D. (1983) “Blind signatures for intraceable payments”
[2] NSA (1996) “How to make a Mint: the cryptography of anonimous electronic cash”
[3] Satoshi Nakamoto (2008): “Bitcoin: a peer-to-peer electronic cash system”
[4] A. Narayanan, J. Bonneau, E. Felten, A. Miller, S. Goldfeder: “Bitcoin and Cryptocurrency Technologies”
https://d28rh4a8wq0iu5.cloudfront.net/bitcointech/readings/princeton_bitcoin_book.pdf
[5] A. Aantonopulos:“Mastering Bitcoin”
https://github.com/bitcoinbook/bitcoinbook/blob/develop/book.asciidoc
[6] https://bitcoin.org/en/developer-guide#transactions
[7] http://davidederosa.com/basic-blockchain-programming/
[27] https://github.com/**bitcoinj/bitcoinj**
[8] https://www.interlogica.it/insight/il-consenso-di-nakamoto/
[9] J.W. Bos et al.(2013): “Elliptic Curve Cryptography in Practice”
https://eprint.iacr.org/2013/734.pdf
[10] https://bitcointalk.org/index.php?topic=1339031.0
[11] https://it.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm
[25] https://www.coindesk.com/math-behind-bitcoin/
[26] https://www.bbc.com/news/technology-12116051

2. Inalterabilità del libro mastro ….....

[12] S. Haber, W.S. Stornetta(1991) “How to time-stamp a digital document”
[13] R. Merkle (1979)"Method of providing digital signatures" US patent 4309569
[14] C.Duark, M.Naor “Pricing via processing, or, combatting junk mail” 1992
[15] Wei Dai (1998) “B-money”
[16] N. Szabo (2005) “Bitgold”
[17] http://www.treccani.it/enciclopedia/rovina-del-giocatore_%28Dizionario-di-Economia-e-Finanza%29/

3. Il controllo delle transazioni ….....

[18] https://www.paranoidbitcoin.com/2017/09/12/a-tuesday-two-part-2-p2pkh-vs-p2sh/
[19] https://github.com/bitcoin/bips/blob/master/bip-0016.mediawiki

4. La sicurezza delle chiavi ….....

[20] https://www.bitaddress.org/
[21] https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki
[22] Robert Morris, Ken Thompson (1978):“Password Security: A Case History”
[23] Shamir A.(1979)”How to share a secret”
[24] https://it.wikipedia.org/wiki/Interpolazione_di_Lagrange