Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Matematica

Il processo di Mining e la Blockchain

Tesi di Laurea in Crittografia

Relatore:
Chiar.mo Prof.
DAVIDE ALIFFI

Presentata da:
MARIA D'ILIO

Sessione unica
Anno Accademico 2018-2019

A nonna Maria.

Introduzione

Bitcoin è un sistema di crittovaluta elettronica inventato da Satoshi Nakamoto nel 2008 ed esaminato nel suo articolo dal titolo *"Bitcoin: a peer-to-peer electronic cash sistem"*, in cui spiega come vengono risolti i problemi che insorgono in un sistema non regolamentato da una Banca centrale, bensì da un network di nodi peer-to-peer.
Una delle maggiori difficoltà di questo nuovo sistema è la necessità di eliminare la fiducia verso terze parti durante una transazione: l'obiettivo è far sì che tutti siano d'accordo sull'ordine cronologico delle transazioni, in modo tale da evitare il problema del double-spending.
Per fare ciò è necessario introdurre un algoritmo di consenso che permetta di mettere d'accordo i diversi nodi del network e che permetta di risolvere il cosiddetto "Problema dei Generali Bizantini": si supponga che due generali, dislocati in diverse aree geografiche, debbano decidere come coordinare l'attacco di una città nemica, potendo comunicare solo mediante messaggeri. Tra questi è altamente probabile che vi siano dei traditori che mandino messaggi che vanno contro la strategia dell'esercito. Il problema è portare avanti l'attacco in modo efficiente nonostante il rischio di tradimento: ciò è conosciuto come consenso decentralizzato.
Nakamoto risolse questo problema grazie alla catena delle proof-of-work basata sulla potenza computazionale dei nodi.

Il seguente elaborato di tesi mira ad analizzare i meccanismi alla base del sistema Bitcoin, focalizzandosi sull'aspetto crittografico.

Nel primo capitolo si riportano le nozioni di crittografia che saranno alla base della Blockchain e delle crittovalute, cercando di dare le nozioni di base per comprendere il sistema crittografico sul quale si basano le transazioni, ossia l'elliptic Curve DSA.
Si studiano poi le transazioni, la loro struttura e lo scambio di denaro basato sugli UTXO (Unspent Transaction Outputs) e si conclude analizzando la Blockchain ed il Mining, cioè il meccanismo grazie al quale le operazioni vengono validate.

Indice

Elenco delle figure

Capitolo 1

Alla base della Blockchain: Crittografia a Chiave Pubblica e a Curve Ellittiche

La proprietà dei bitcoin si stabilisce attraverso *chiavi digitali*, *indirizzi bitcoin* e *firme digitali*.
Le chiavi private non sono salvate nella rete, ma sono create e memorizzate in un semplice database chiamato *wallet* dagli utenti, i quali le gestiscono senza relazione con la blockchain.
Ogni transazione bitcoin richiede una firma valida per essere inclusa nella blockchain, che può essere generata solo con chiavi digitali valide; controllando le transazioni sulla blockchain (accessibile a tutti) il wallet può fare il conto dei bitcoin di un determinato utente.
In una transazione bitcoin, la chiave pubblica del destinatario è rappresentata dalla sua impronta digitale, chiamata "indirizzo bitcoin", che ha la stessa funzione del nome del beneficiario su un assegno. Nella maggior parte dei casi un indirizzo bitcoin è generato a partire da una chiave pubblica, a cui corrisponde ed è l'unica rappresentazione delle chiavi che gli utenti regolarmente vedono.
La crittografia a chiave pubblica è stata inventata negli anni '70 ed è una

delle basi della sicurezza informatica.

Sono state scoperte diverse funzioni matematiche, come l'elevazione a potenza dei numeri primi e la moltiplicazione su una curva ellittica, facili da calcolare in una direzione, ma difficili da invertire; di quest'ultima si serve il sistema Bitcoin come base per la sua crittografia a chiave pubblica.

In questo capitolo si daranno delle nozioni di crittografia che saranno lo strumento di base per capire il funzionamento del Mining, processo alla base della Blockchain e delle criptovalute.

1.1 Definizioni preliminari

Prima di dare le nozioni fondamentali di questa trattazione, è necessario formalizzare cosa si intenda per un algoritmo eseguire una funzione.

Definizione 1.1.1 (Algoritmo)**.** Un algoritmo è una *Macchina di Turing* deterministica che ha in input e output stringhe sull'alfabeto $\Sigma = \{0, 1\}$.

Definizione 1.1.2 (Tempo di esecuzione)**.** Un algoritmo A si dice che esegue una funzione in un tempo $T(n)$ se per ogni $x \in \{0,1\}^*$, $A(x)$ termina al massimo in $T(\mid x \mid)$ passi. Si dice che A termina in un tempo polinomiale se esiste una costante c tale che $A(x)$ termina in $T(\mid x \mid) = \mid x \mid^c$.

Definizione 1.1.3 (Deterministic Computation)**.** Un algoritmo A calcola in modo deterministico una funzione $f\colon \{0,1\}^* \to \{0,1\}^*$ se per ogni $x \in \{0,1\}^*$, dato ad A x in input, restituisce come output $f(x)$.

Diciamo che un algoritmo è efficiente se termina in un tempo polinomiale.

Una naturale estensione del caso deterministico è assumere che un algoritmo in alcuni casi 'getti una moneta', (utilizzi un generatore di bit casuali) generando così un nastro di bit casuali.

Definizione 1.1.4 (Tempo di esecuzione)**.** Si dice che un algoritmo *p.p.t.* A esegue una funzione in un tempo $T(n)$ se per ogni $x \in \{0,1\}^*$ e per ogni nastro random $A(x)$ termina al massimo in $T(\mid x \mid)$ passi. Si dice che A termina in un tempo polinomiale se esiste una costante c tale che $A(x)$ termina in $T(\mid x \mid) = \mid x \mid^c$

Dobbiamo dunque estendere la nostra nozione di calcolo di una funzione nel caso di un algoritmo probabilistico.
In generale diremo che se un algoritmo è probabilistico allora i suoi output diventano una distribuzione su una famiglia di stringhe sull'alfabeto $\Sigma = \{0,1\}$.

Definizione 1.1.5 (Algoritmo Random (*p.p.t.*))**.** Un algoritmo random, chiamato anche Probabilistic Polynomial Time Turing machine (*p.p.t.*), è una macchina di Turing dotata di un nastro casuale supplementare. Ogni bit del nastro casuale viene scelto uniformemente e in modo indipendente.

Ugualmente importante è il concetto di *funzione unidirezionale*.
Dato un sistema crittografico

- deve essere semplice calcolare il testo cifrato c dati il messaggio m e la chiave k
- è difficile recuperare m e k dato c.

Queste due richieste suggeriscono l'uso di funzioni che sono facili da calcolare, ma difficili da invertire, ovvero *easy to compute but hard to invert*: le funzioni unidirezionali (OW dall'inglese "One Way").
Si hanno tre diverse definizioni di funzione OW; la prima, nonchè quella che formalizza le nostre intuizioni in un modo molto semplice, è la seguente

Definizione 1.1.6. Una funzione $f : \{0,1\}^* \to \{0,1\}^*$ è una funzione One Way nel caso peggiore (worst case OW function) se è:

- **Easy to compute.** c'è un algoritmo *p.p.t.* C che calcola $f(x)$ per tutti gli input $x \in \{0,1\}^*$

- **Hard to invert.** non ci sono avversari A tali che

$$\forall x \in \{0,1\}^* \quad Pr[A(f(x)) \in f^{-1}(f(x))] = 1$$

Questa definizione considera funzioni OW solo quelle per cui non esiste un avversario A che inverte $f(x)$ per tutti gli x; tale condizione risulta difficilmente verificabile, motivo per il quale si è soliti utilizzare definizioni con condizioni leggermente meno restrittive.

Definizione 1.1.7 (Funzioni OW in senso Forte)**.** Una funzione $f : \{0,1\}^* \to \{0,1\}^*$ è una funzione OW in senso forte se è:

- **Easy to compute.** c'è un algoritmo *p.p.t.* C che calcola $f(x)$ per tutti gli input $x \in \{0,1\}^*$
- **Hard to invert.** ogni tentativo efficiente di invertire f ha successo al massimo con una probabilità trascurabile. Formalmente, per ogni avversario A, esiste una funzione trascurabile $\epsilon(\cdot)$ tale che per ogni input di lunghezza $n \in \mathbb{N}$

$$Pr[x \longleftarrow \{0,1\}^n; y \longleftarrow f(x) \colon f(A(1^n, y)) = y] \leq \epsilon(n)$$

  con $\epsilon(\cdot)$ asintoticamente trascurabile.

Sfortunatamente, non tutte le funzioni naturali candidate ad essere OW soddisfano la seconda condizione: infatti, in certi casi questa deve essere più debole, cosicché risulta essere conveniente introdurre la seguente

Definizione 1.1.8 (Funzioni OW in senso Debole)**.** Una funzione $f : \{0,1\}^* \to \{0,1\}^*$ è una funzione OW in senso debole se è:

- **Easy to compute.** c'è un algoritmo *p.p.t.* C che calcola $f(x)$ per tutti gli input $x \in \{0,1\}^*$
- **Hard to invert.** esiste una funzione polinomiale $q : \mathbb{N} \to \mathbb{N}$ tale che per ogni avversario A, per un $n \in \mathbb{N}$,

$$Pr[x \longleftarrow \{0,1\}^n; y \longleftarrow f(x) \colon f(A(1^n, y)) = y] \leq 1 - \frac{1}{q(n)}$$

In generale una funzione OW $f(\cdot)$ non garantisce che il primo bit (o equivalentemente un qualsiasi bit fissato dell'input) sia difficile da calcolare dati gli output.
Tuttavia, se questo bit è difficile da calcolare è chiamato *bit Hard Core*.
Si dia ora una definizione formale

Definizione 1.1.9 (Bit Hard Core)**.** Una funzione $h\colon \{0,1\}^n \to \{0,1\}$ è Hard Core per $f(x)$ se è facile da calcolare dato x ma per ogni avversario *p.p.t.* A esiste una funzione trascurabile $\epsilon(\cdot)$ tale che $\forall k \in \mathbb{N}$

$$Pr[x \longleftarrow \{0,1\}^k : A(1^n, f(x)) = h(x)] \leq \frac{1}{2} + \epsilon(n)$$

Dunque, il predicato Hard Core è facile da calcolare dato x ma difficile da calcolare dato $f(x)$.

Osservazione 1.1.1. Si può dimostrare che ogni funzione OW ha almeno un predicato *Hard Core*.

1.2 Crittografia a Chiave Pubblica

In un sistema a chiave pubblica si hanno due chiavi:

- una chiave privata per decifrare (**sk**)
- una chiave pubblica per cifrare (**pk**)

La chiave pubblica è inserita in un "ripostiglio" sicuro, in modo tale da permettere a tutti di utilizzarla per cifrare i messaggi, mentre la chiave privata, al contrario, si tiene segreta ed è utilizzata solamente da colui che deve decifrare il messaggio.

Si definisce un sistema di cifratura a chiave pubblica come segue:

Definizione 1.2.1 (Public Key Encryption Scheme)**.** Una tripla di algoritmi (Gen, Enc, Dec) è un sistema di cifratura a chiave pubblica se

1. $(pk, sk) \longleftarrow Gen(1^n)$ è un algoritmo *p.p.t.* che produce la coppia delle chiavi (pk, sk)
2. $c \longleftarrow Enc_{pk}(m)$ è un algoritmo *p.p.t.* che dato pk restituisce $c \in \{0,1\}^n$
3. $m \longleftarrow Dec_{sk}(c)$ è un algoritmo deterministico che dato in input un testo cifrato c e una chiave segreta sk produce un messaggio $m \in \{0,1\}^n \cup \perp$
4. Per ogni $n \in \mathbb{N}, m \in \{0,1\}^n$

$$Pr[(pk, sk) \longleftarrow Gen(1^n) : Dec_{sk}(Enc_{pk}(m)) = m] = 1$$

Si noti che nella definizione è stato inserito il simbolo $\perp$ per indicare che il messaggio da decodificare risulta "indecifrabile".

Definizione 1.2.2 (Secure Public Key Encription)**.** Un sistema a chiave pubblica (Gen,Enc,Dec) è detto *sicuro* se per ogni algoritmo D *p.p.t.* esiste una funzione trascurabile $\epsilon(\cdot)$ che per ogni $n \in \mathbb{N}$, per ogni $m_0, m_1 \in \{0,1\}^n$, D distingue le distribuzioni

- $\{(pk, sk) \longleftarrow Gen(1^n) \colon (pk, Enc_{pk}(m_0))\}_n$
- $\{(pk, sk) \longleftarrow Gen(1^n) \colon (pk, Enc_{pk}(m_1))\}_n$

con probabilità al massimo $\epsilon(n)$.

Analizziamo ora l'**ElGamal Cryptosystem** basato sul **Problema del logaritmo discreto**.

Osservazione 1.2.1 (Ipotesi del Logaritmo Discreto)**.** Se G_q è un gruppo con ordine primo q, allora per ogni avversario A $\exists\epsilon(\cdot)$ funzione trascurabile tale che

$$Pr[q \longleftarrow \Pi_n; g \longleftarrow < G_q >; x \longleftarrow \mathbb{Z}_q : A(g^x) = x] < \epsilon(n)$$

Con $< G_q >$ insieme di generatori del gruppo G_q e Π_n insieme di primi rappresentabili con n bit.

In crittografia assumendo l'ipotesi del Logaritmo Discreto è improbabile trovare la soluzione del logaritmo discreto, ma l'operazione inversa, ossia l'esponenziazione può essere calcolata in modo efficiente.

Crittosistema 1.2.1 (Schema di crittografia a chiave Pubblica di **ElGamal**)**.** *Preso un primo p tale che valga l'Ipotesi del logaritmo discreto su $(\mathbb{Z}_p^*, \cdot)$, sia $\alpha \in \mathbb{Z}_p^*$.*
Si definisca

$$K = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod p\}$$

I valori p, α e β costituiscono la chiave pubblica , mentre a la chiave privata; cioè $pk = (p, \alpha, \beta)$ e $sk = (a)$.
Per K= (p, α, a, β) e per un numero (segreto) random $k \in \mathbb{Z}_{p-1}$,
si ha

$$Enc_{pk}(x, k) = (y_1, y_2)$$

con

$$y_1 = \alpha^k \pmod p$$

e

$$y_2 = x\beta^k \pmod{p}$$

Si ha dunque

$$Dec_{sk}(y_1, y_2) = y_2(y_1^a)^{-1} \pmod{p}$$

1.3 Firma digitale

Si supponga ora che Bob riceva un messaggio da Alice. Come fa Bob ad essere sicuro della provenienza del messaggio?
In questa sezione si analizzerà uno dei metodi che rendono difficile il "forzare una firma": la *firma digitale*; questa è uno schema matematico utile a dimostrare l'autenticità di un messaggio o di un documento digitale. Una firma digitale valida consente a un destinatario di verificare che il messaggio sia stato creato da un mittente noto, cosicché il mittente non possa negarne l'invio e il messaggio non possa essere modificato durante il transito. E' possibile usare le firme digitali in un sistema a chiave pubblica.

Definizione 1.3.1 (Schema con Firma Digitale)**.** (Gen,Sign,Ver) è uno *schema con firma digitale sullo spazio dei messaggi* $\{M_n\}_n$ se

- $Gen(1^n)$ è un algoritmo *p.p.t.* che dato n restituisce in output una chiave pubblica pk e una chiave segreta sk: $(pk, sk) \longleftarrow Gen(1^n)$.
- $Sign$ è un algoritmo *p.p.t.* che presi in input la chiave privata sk e il messaggio m restituisce una firma σ: $\sigma \longleftarrow Sign_{sk}(m)$.
- Ver è un algoritmo *p.p.t.* deterministico che presi in input la chiave pubblica pk, il messaggio m e la firma σ restituisce "accept" o "reject".
- $\forall n \in \{0,1\}^n, \forall m \in M_n$,

$$Pr[(pk, sk) \longleftarrow Gen(1^n) : Ver_{pk}(m, Sign_{sk}(m)) ='' accept''] = 1$$

La sicurezza di una firma digitale può essere definita considerando un avversario che può fare un numero polinomiale di domande ad un oracolo di firma (rappresentato con $A^{Sign_{sk}(\cdot)}$). Non è considerata una falsificazione se l'avversario A produce una firma su un messaggio m su cui ha interrogato l'oracolo di firma. Nota che per definizione di sistema a chiave pubblica, l'avversario ha libero accesso all'oracolo dell'algoritmo di verifica Ver_{pk}.

Definizione 1.3.2 (Sicurezza della Firma Digitale)**.** (Gen,Sign,Ver) è *sicuro* se per ogni avversario *p.p.t.* A esiste una funzione trascurabile $\epsilon(\cdot)$ tale che $\forall n \in \mathbb{N}$

$$Pr[(pk, sk) \longleftarrow Gen(1^n); (m, \sigma) \longleftarrow A^{Sign_{sk}(\cdot)}(1^n) : \\ A \text{ non chiede } m \wedge Ver_{pk}(m, \sigma) = \text{"}accept\text{"}] \leq \epsilon(n)$$

Costruiamo ora un **Sistema efficiente con Firma Digitale** utilizzando le *trapdoor OW function*, ovvero funzioni OW alle quali però si aggiunge come condizione quella di essere invertibili, data un'informazione aggiuntiva.
Si ha

- $Gen(1^n)$: $pk = i$ e $sk = t$ la *trapdoor*.
- $Sign_{sk}(m) = f^{-1}(m)$ utilizzando t
- $Ver_{pk}(\sigma, m) =$ "accept" se $f_i(\sigma) = m$

Questo schema però non è sicuro se l'avversario può scegliere il messaggio da contraffare.
Presentiamo ora lo schema di ElGamal nella versione di firma, descritto nel 1985; questo schema venne successivamente modificato nel Digital Signature Algorithm (o DSA) dal National Institute of Standard and Tecnology.
Lo schema di ElGamal nella versione di firma (chiamato anche ElGamal Pulic-key Cryptosistem) è non deterministico: questo significa che ci sono più firme valide per ogni messaggio dato e che la verifica dell'algoritmo deve essere in grado di accettare ognuna di queste come autentiche.

Analizziamo ora il

Crittosistema 1.3.1 (**ElGamal Signature Scheme**)**.** *Sia p un primo tale per cui il problema del logaritmo discreto in $\mathbb{Z}_p$ è intrattabile, sia $\alpha \in \mathbb{Z}_p^*$. Preso $p \in \mathbb{Z}_p^*, A = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1}$, definiamo*

$$K = \{(p, \alpha, a, \beta) : \beta) \equiv \alpha^a \pmod{p}\}$$

I valori p, α e β costituiscono la chiave pubblica, mentre a rappresenta la chiave privata; cioè: pk=(p,α, β) e sk=(a).
Per K= (p, α, a, β) e per un numero (segreto) random $k \in \mathbb{Z}^*_{p-1}$, si definisce

$$\boldsymbol{sig}_K(x, k) = (\gamma, \delta)$$

con

$$\gamma = a^k \pmod{p}$$

e

$$\delta = (x - a\gamma)k^{-1} \pmod{p-1}$$

Per $x, \gamma \in \mathbb{Z}_{p-1}$, definiamo

$$\boldsymbol{ver}_K(x, (\gamma, \delta)) = true \iff \beta^\gamma \gamma^\delta \equiv \alpha^x \pmod{p}$$

Esempio 1.3.1. Si prenda un primo $p = 467, \alpha = 2, a = 127$; si prenda

$$\beta = \alpha^a \pmod{p} = 2^1 27 \pmod{467} = 132$$

Supponiamo che Alice voglia inviare un messaggio $x = 100$, scegliendo un $k = 213$ (si noti che $gcd(213, 466) = 1$)
si ha
$k^{-1} \pmod{p-1} = 213^{-1} \pmod{466} = 431$.
Allora si calcoli

$$\gamma = \alpha^k \pmod{p} = 2^2 13 \pmod{467} = 29$$

e

$$\delta = (x - a\gamma)k^{-1} \pmod{p-1} = (100 - 127 \cdot 29)431 \pmod{466} = 51$$

Dunque, si ha che

$$\beta^\gamma \gamma^\delta \equiv 132^{29} \cdot 29^{51} \pmod{469} = 189$$

e

$$\alpha^x \pmod{p} \equiv 2^{100} \pmod{467} = 189$$

1.4 Crittografia a Curve Ellittiche

La Crittografia ellittica è una tipologia di crittografia basata sulle curve ellittiche in un campo finito.

Siano $\mathbb{K}$ un campo e $f : \mathbb{K} \times \mathbb{K} \to \mathbb{K}$ una funzione della forma

$$f(x, y) = y^2 - x^3 - ax - b$$

dove $a, b, x, y \in \mathbb{K}$ ed f è non singolare.

Definizione 1.4.1 (Curva Ellittica)**.** Una curva ellittica **E** su un campo $\mathbb{K}$, che sarà denotata con $\mathbf{E}/\mathbb{K}$, o semplicemente con **E**, è l'insieme dei punti $\{(x, y) \in \mathbb{K} \times \mathbb{K} \colon f(x, y) = y^2 - x^3 - 2 - ax - b = 0\}$ tale che $4a^3 + 27b^2 \neq 0$, più un punto $\mathcal{O}$ che chiameremo "punto all'infinito".
Dunque, si ha

$$\mathbf{E} = \{(x, y) \in \mathbb{K} \times \mathbb{K} \colon y^2 = x^3 + ax + b, 4a^3 + 27b^2 \neq 0\} \cup \{\mathcal{O}\}$$

Esempio 1.4.1. Esempio di curva ellittica su $\mathbb{R} \times \mathbb{R}$.
Consideriamo **E**: $y^2 = x^3 - 4x$

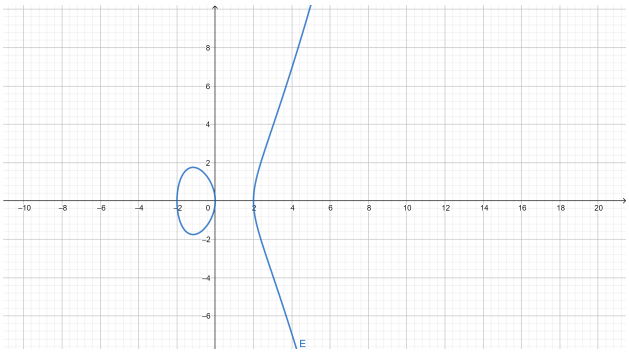

Figura 1.1: Esempio di curva ellittica **E**: $y^2 = x^3 - 4x$

Osservazione 1.4.1. $4a^3 + 27b^2 \neq 0$ è la condizione di non singolarità e assicura che la curva presa in esame non sia singolare, ovvero senza punti

singolari.
Un punto $P = (x, y) \in \mathbb{K} \times \mathbb{K}$ è detto **punto singolare** se

$$\frac{\partial f}{\partial x}(P) = \frac{\partial f}{\partial y}(P) = 0$$

Esempio 1.4.2. Esempio di curva su $\mathbb{R} \times \mathbb{R}$ con un punto singolare

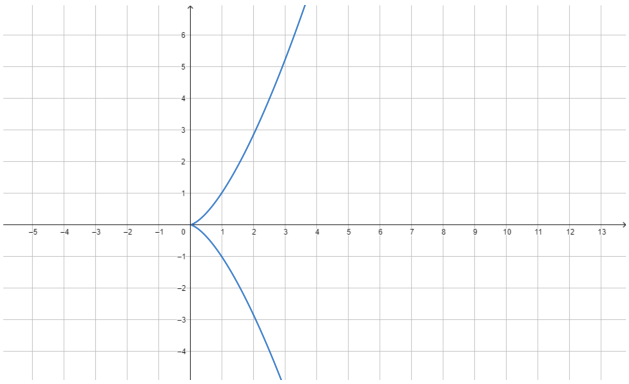

Figura 1.2: Esempio di curva singolare **E**: $y^2 = x^3$

Una delle caratteristiche dell'insieme di punti di una curva ellittica è che è sempre possibile definire una struttura algebrica su di essa che è data da un'operazione di addizione rispetto alla quale essa è un gruppo abeliano, generalmente utilizzato per definire i crittosistemi.
Nel caso reale, per addizionare due punti della curva ellittica P e Q è necessario solamente prendere in considerazione la retta che passa per i due punti che intersecherà **E** in un terzo punto R'. La somma sarà il simmetrico di R' su **E**, che chiameremo R.

Osservazione 1.4.2. Se $\mathbb{K}$ è un campo finito allora **E** ha un numero finito di punti.

Esempio 1.4.3. In $\mathbb{Z}_p$, con $p = 11$.
Consideriamo **E** : $y^2 = x^3 + x + 6$.
Nella prima colonna della tabella seguente si hanno i valori che la x può assumere, nelle colonne seguenti si trovano i valori di y^2 e successivamente,

dopo aver controllato che esso sia effettivamente un quadrato, si riportano i valori di y e i conseguenti punti della curva ellittica **E**.

x	y^2	y	punti
0	6 (mod 11)		
1	8 (mod 11)		
2	5 (mod 11)	4;7	A;B
3	3 (mod 11)	5;6	C;D
4	8 (mod 11)		
5	4 (mod 11)	2;9	E;F
6	8 (mod 11)		
7	4 (mod 11)	2;9	G;H
8	9 (mod 11)	3;8	I;J
9	7 (mod 11)		
10	4 (mod 11)	2;9	K;L

Nel grafico seguente riportiamo i punti della curva ellittica **E** appena analizzata.

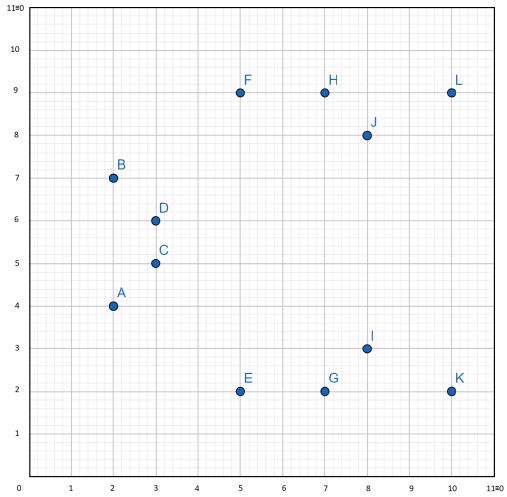

Figura 1.3: Curva ellittica **E** sul campo $\mathbb{Z}_{11} \times \mathbb{Z}_{11}$

Ovviamente a questo grafico si aggiunge il "punto all'infinito"$\mathcal{O}$ che ha il ruolo dello 0 nell'addizione[1].

Come si vede dall'esempio, poiché la curva è definita su un campo finito di caratteristica un primo p invece che su campo reale, la curva ellittica è rappresentata da un insieme di punti in due dimensioni: la matematica utilizzata è però la stessa.

Per visualizzare la moltiplicazione di un punto sulla curva ellittica per un intero viene utilizzata per semplicità una curva ellittica sul campo reale, ricordando che la matematica utilizzata è la stessa.

Si traccia la tangente della curva ellittica nel punto A, l'intersezione tra la tangente e la curva sarà -2A; il punto 2A sarà la riflessione di -2A rispetto all'asse x.

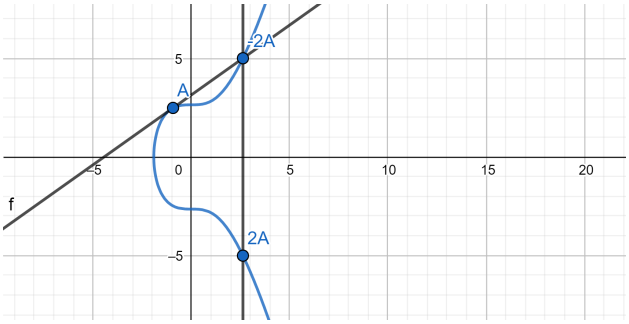

Figura 1.4: Moltiplicazione di un punto A per un intero k=2 su una curva ellittica

[1]In geometria proiettiva è il punto che si rappresenta come [0,1,0]

1.5 Chiavi e Indirizzi Wallet

Un portafoglio Bitcoin contiene una lista di coppie di chiavi, composte da una chiave pubblica e una privata; da quest'ultima, utilizzando la crittografia a curva ellittica, si ricava la chiave pubblica e di conseguenza, mediante una funzione di hash crittografica, si ottiene l'indirizzo bitcoin.

Una funzione crittografica di hash

$$h\colon \{0,1\}^* \to \{0,1\}^n$$

è un algoritmo matematico che, presi dei dati di lunghezza arbitraria, restituisce una stringa binaria di dimensione fissata chiamato valore di hash o anche digest.
La funzione di hash è progettata in modo che:

- $h(m)$ è calcolabile in modo efficiente
- unidirezionale: dato $y \in \{0,1\}^n$ è impossibile trovare $m \in \{0,1\}^*$ tale che $h(m) = y$
- resistenza alle collisioni: $\forall m_1, m_2 \in \{0,1\}^*, h(m_1) \neq h(m_2)$

Proprio queste proprietà assicurano la sicurezza del sistema.

Una **chiave privata** (sk) è semplicemente un numero scelto arbitrariamente, la cui conoscenza è alla base del controllo dell'utente su tutti i fondi associati all'indirizzo bitcoin corrispondente ed è utilizzata per creare firme che sono necessarie per trasferire bitcoin da un indirizzo ad un altro dimostrando così la proprietà dei fondi utilizzati nella transazione. Questa ovviamente deve rimanere segreta in ogni momento e deve essere conservata per evitare un eventuale smarrimento, cosa che comporterebbe la perdita di tutti i fondi ad essa associati.
Essa è un numero compreso tra 1 e $n-1$, in cui n è una costante definita come l'ordine della curva ellittica usata in bitcoin. Per creare la chiave, si

sceglie casualmente un numero a 256-bit e inferiore a $n - 1$; per fare ciò si genera una stringa più lunga di bit random e tramite un algoritmo di hash SHA256, si produrrà un numero da 256-bit. Se il risultato è minore di n-1, si avrà una chiave privata adeguata, altrimenti si ripete il procedimento.

La **chiave pubblica** (pk) è calcolata dalla chiave privata utilizzando la proprietà di moltiplicazione delle curve ellittiche, che è un'operazione irreversibile: $pk = sk * G$ con sk chiave privata e G punto della curva costante chiamato "punto di generazione".
Il punto G risulta essere unico per tutti gli utenti bitcoin, quindi una stessa chiave sk genererà una stessa chiave pk. Si nota dunque che la relazione tra pk e sk è fissata ma può essere calcolata solo in una direzione, da sk a pk, poichè la funzione matematica è unidirezionale.

L'**indirizzo bitcoin** può essere condiviso con chiunque voglia inviarvi denaro; esso si ricava dalla chiave pubblica e consiste in una stringa di lettere e numeri che inizia con la cifra "1" corrispondente al destinatario dei fondi; si ricava dunque dalla chiave pubblica mediante l'uso di una funzione di hash. In modo particolare vengono usate in sequenza SHA256 e RIPEDEM160, producendo così un numero a 160 bit solitamente presentato in una codifica chiamata "Base58Check" che utilizza 58 caratteri in modo tale da aumentare la leggibilità da parte degli utenti.

Capitolo 2

Le Transazioni

Il sistema Bitcoin, al contrario dei convenzionali sistemi bancari e di pagamento, è basato sulla *decentralizzazione della fiducia*, grazie alla quale, anziché da un'autorità centrale fiduciaria, la fiducia si ottiene dall'interazione di diversi partecipanti nel sistema bitcoin.
Il problema fondamentale di questo sistema è ovviamente quello di autenticare una transazione e impedire il double-spending (doppia spesa).

Le transazioni sono la parte più importante del sistema bitcoin che è costruito in modo tale da assicurare la creazione della valuta, la propagazione nel network, la validazione e la sua aggiunta al registro globale delle transazioni: la **Blockchain**.

Una transazione, dopo la sua creazione (anche detta originazione) deve essere firmata per permettere la spesa dei fondi registrati in essa e viene successivamente trasmessa nel network bitcoin, validata da ogni suo nodo finché non viene propagata nell'intera rete. Questa viene quindi verificata da un nodo miner ed inclusa in un blocco di transazioni registrato nella blockchain; infine, viene registrata nella blockchain e confermata da un numero di blocchi successivi sufficienti ed è contabilizzata perennemente nel registro bitcoin.

La firma convalida la transazione permettendole così di contenere tutte

le informazioni necessarie per il trasferimento dei fondi.
Non c'è bisogno di fiducia reciproca tra nodi e mittente, poiché la transazione è firmata e non contiene dati sensibili o confidenziali: essa può essere trasmessa in reti non sicure affinché il trasporto dei dati sia conveniente. Questa è una grande differenza dalle transazioni eseguite mediante carte di credito, poiché quest'ultime contengono dati sensibili e possono essere trasmesse solo su reti protette.
Una volta che una transazione è inviata a qualsiasi nodo connesso alla rete bitcoin, quella transazione sarà validata da quel nodo; se quest'ultimo la ritiene valida, la invierà ai nodi a cui è connesso e un messaggio di conferma arriva all'originatore; un meccanismo simile avviene se il nodo non la ritiene valida: rifiuta la transazione e invia un messaggio di rigetto all'originatore.

La rete bitcoin è una rete ***peer-to-peer***: questo vuol dire che ogni nodo è connesso con altri nodi bitcoin che vengono scoperti durante l'avvio del programma, attraverso il protocollo *peer-to-peer*; l'intera rete risulta perciò essere una sorta di maglia senza una struttura ben definita, rendendo tutti i nodi peer alla pari. I messaggi, includendo transazioni e blocchi, sono propagati attraverso un processo chiamato "flooding"(inondazione): quando il messaggio arriva ad un nodo, questo lo invia ai nodi adiacenti e così via. In questo modo, una transazione valida si propagherà esponenzialmente attraverso il network finché tutti i nodi non l'avranno ricevuta, ma ogni nodo controlla in modo indipendente la validità della transazione.
La rete bitcoin è progettata per propagare transazioni e blocchi a tutti i nodi in modo efficiente e resiliente, ovvero resistente agli attacchi, per prevenire i quali ogni nodo valuta in modo indipendente ogni transazione prima di divulgarla ulteriormente; infatti, una transazione corrotta non riuscirà a superare un nodo.

2.1 Struttura delle Transazioni

Una transazione è una struttura di dati che codifica e trasferisce un valore da una fonte di fondi, chiamata *input*, ad una destinazione, chiamata *output*. Gli input e gli output sono relazionati ad account o identità; si dovrebbero immaginare come somme di bitcoin impacchettati e protetti tramite una chiave segreta conosciuta solo dal proprietario.
Una transazione è formata da un certo numero di campi:

DIMENSIONE	CAMPO	DESCRIZIONE
4 bytes	Versione	Specifica quale regola segue la transazione
1-9 bytes	Input counter	Quanti input sono inclusi
Variable	Inputs	Uno o più input di transazione
1-9 bytes	Output counter	Quanti output sono inclusi
Variable	Outputs	Uno o più output di transazione
4bytes	Locktime	Numero del blocco

Osservazione 2.1.1. Si definisce **Locktime** il primo tempo in cui una transazione è considerata valida e può essere trasmessa alla rete o aggiunta alla blockchain; è impostato come zero in molte transazioni per richiedere la propagazione immediata: se è diverso da zero ma minore di 500 milioni, è interpretato come un block height, cioè si intende che la transazione non è valida e non sarà inclusa nella blockchain prima di un certo blocco. Se è maggiore di 500 milioni, è interpretato come ”numero di secondi dal 1 gennaio 1970” e la transazione non sarà valida prima del tempo specificato.
L'uso del locktime è equivalente al postdatare un assegno.

2.2 Input e Output di una Transazione

Gli elementi fondamentali di una transazione bitcoin sono gli *output di transazione non spesi* (Unspent Transaction Outputs), o UTXO, ovvero frammenti di valuta bitcoin indivisibili di uno specifico proprietario, registrati nella blockchain e riconosciuti come validi da tutto il network.
La rete traccia tutti gli UTXO non spesi: quando un utente riceve bitcoin, quella somma diventa input di una transazione ed è registrata nella blockchain. Poiché non si hanno bilanci in bitcoin, ma solo *output di transazione non spesi* (UTXO) sparsi nella blockchain, i bitcoin di un utente potrebbero essere distribuiti nel network in UTXO in transazioni e blocchi. Non c'è un saldo dei vari indirizzi, ma questo compito è gestito dai wallet analizzando e scansionando la blockchain.
Un UTXO può avere un valore arbitrario denominato in multipli di **satoshi**, divisione del bitcoin all'ottava cifra decimale. Gli UTXO possono contenere un qualsiasi valore, ma una volta creati sono indivisibili. Se un UTXO è più grande del valore desiderato per eseguire una transazione, deve essere comunque utilizzato nella sua interezza e deve essere generato un resto della transazione.

Esempio 2.2.1. Si immagini di dover acquistare un bene da 2,00€, prendendo il portafoglio si cerca di trovare una combinazione di monete e banconote per coprire la quantità di denaro necessaria. Allo stesso modo, una transazione bitcoin è formata da più UTXO combinati in modo tale da comporre una somma più grande o uguale al valore di transazione desiderato.
Come nella vita reale, l'applicazione bitcoin può usare diverse strategie: combinare quantità più piccole o usare unità più grandi del valore desiderato; in entrambi i casi c'è la necessità della creazione del resto.

Tutto ciò è gestito automaticamente dal wallet dell'utente, tenendo un database degli UTXO bloccati, in un processo invisibile all'utente.
Gli UTXO consumati nella transazione costituiscono gli *input della transazione* (transaction input) e gli UTXO creati dalla transazio-

ne costituisco gli *output della transazione* (transaction output). In questo modo una parte di valore bitcoin si muove in avanti da un proprietario all'altro formando una catena di transazioni che consumano e creano UTXO. Le transazioni consumano UTXO sbloccandoli con la firma del proprietario attuale e creano UTXO bloccandoli sull'indirizzo bitcoin di un nuovo proprietario.

Tenendo presente che c'è sempre differenza tra l'input e l'output totale, data dalle commissioni di rete, ogni transazione è composta da uno o più output e uno o più input.

Gli **output** di una transazione consistono in due parti:

- un importo di bitcoin in satoshi (1 satoshi= 10^{-8} bitcoin)
- un *locking script* che blocca l'importo specificando le condizioni che devono essere soddisfatte per permettere la spesa; formato da firma digitale e da altre condizioni da soddisfare per spendere la quantità di denaro

Nell'output di una transazione, si ha

DIMENSIONE	CAMPO	DESCRIZIONE
8 bytes	Importo	Valore bitcoin in satoshi
1-9 bytes	Dimensione del Locking-Script	Lunghezza del Locking-Script in byte
Variable	Script di Locking	Uno script che definisce le condizioni necessarie per spendere l'output

Gli **input** di transazione identificano quali UTXO saranno spesi e forniscono la prova della proprietà tramite lo script di sblocco. Per costruire una

transazione, un wallet seleziona tra gli UTXO disponibili quelli con valore maggiore o uguale ad un importo da pagare; per ogni UTXO da utilizzare, si genera un puntatore che indica l'UTXO e lo sblocca.
Nell'input di una transazione, si ha

DIMENSIONE	CAMPO	DESCRIZIONE
32 bytes	Hash della transazione	Puntatore alla transazione contenente l'UTXO che andrà speso
4 bytes	Indice dell'Output	Il numero indice dell'UTXO speso; il primo è 0
1-9 bytes	Dimensione dell'Unlocking-Script	Lunghezza dell'Unlocking-Script in bytes
Variable	Unlocking-Script	Uno script che verifica le condizioni del locking script dell'UTXO

2.3 Transaction Fee

Molte transazioni includono delle ***Commissioni di transazione*** che compensano i *minatori bitcoin* (chiamati Miners).
Le Transaction fees servono da incentivo per includere (minare) una transazione nel blocco successivo ed anche come disincentivo contro gli "spam" o qualsiasi altro tipo di abuso del sistema, incoraggiando un costo su ogni transazione. Le spese delle operazioni vengono accreditate al minatore che mina il blocco e le registra sulla blockchain a suo favore.
Le Transaction Fees vengono calcolate in base alla grandezza in kilobytes della transazione, non in base al valore della transazione in bitcoin; il prezzo per kilobyte cambia in base all'andamento del mercato dei minatori, non in base al valore di bitcoin da trasferire.
I miners danno la priorità alle operazioni in base a molti criteri differenti, comprese le commissioni, ma in alcuni casi questi possono agire anche gratuitamente.
E' però vero che le Transaction fees incidono sulla priorità di elaborazione, nel senso che una transazione con commissioni elevate è probabile che venga inserita nel primo blocco prodotto, al contrario di quelle con commissioni basse o assenti.
Dunque le commissioni non sono obbligatorie ma incoraggiano l'elaborazione prioritaria.
Si osserva che le commissioni non possono mai essere superiori ad $\frac{1}{3}$ dell'ammontare dei bitcoin inviati.
In future revisioni del protocollo è previsto che le applicazioni dei wallet utilizzino l'analisi statistica per calcolare in modo più appropriato la commissione da aggiungere in una transazione in base alle commissioni medie di transazioni recenti.

La struttura dei dati delle transazioni non ha un campo per le commissioni, ma le commissioni sono espresse come differenza tra la somma degli input e la somma degli output. Qualsiasi somma in eccesso che rimane dopo che

gli output sono stati detratti da tutti gli input è la commissione che spetta ai minatori.

$$\textbf{Fees} = \text{Sum(Inputs)} - \text{Sum(Outputs)}$$

Osservazione 2.3.1. Le commissioni sono a carico del mittente.

2.4 Una Transazione dal punto di vista Crittografico

Gli step di una transazione bitcoin da un pagante ad un ricevente sono molteplici:

- Il ricevente genera in modo del tutto casuale una chiave privata che viene salvata sul suo computer;
- la chiave privata viene convertita in una chiave pubblica tramite un procedimento basato su un algoritmo chiamato "Elliptic Curve Digital Signature Algorithm" (anche noto come algoritmo ECDSA) che utilizza la curva ellittica $y^2 = x^3 + 7$;
- la chiave pubblica viene ridotta tramite un hash;
- l'hash della chiave pubblica viene ridotto ad un indirizzo di massimo 35 caratteri, che costituisce l'indirizzo del portafoglio del ricevente;
- l'indirizzo viene inviato al pagante;
- il pagante recupera la chiave pubblica corrispondente all'indirizzo;
- il pagante crea la transazione, specificando la sua firma, l'output e il "*signature script*" che include l'hash della chiave pubblica del ricevente;
- il pagante conferma la transazione inviando tutti i bitcoin presenti nell'input in uno o più output;
- la transazione viene trasmessa online dal pagante a tutta la rete;
- i minatori inseriscono nella blockchain la transazione, che viene riconosciuta così dalla comunità;
- il ricevente firma il *signature script* con la propria chiave privata e pubblica, dimostrando così di essere il proprietario dell'output della transazione. Ora può spendere l'output in una nuova operazione.

2.4.1 The Elliptic Curve DSA

Nel 2000 venne approvato l'Elliptic Curve Digital Signature Algorithm(ECDSA), uno schema di firma che può essere visto come uno sviluppo del DSA (Digital Signature Algorithm) nello spazio delle curve ellittiche.
Si analizzi il

Crittosistema 2.4.1 (Digital Signature Algorithm)**.** *Sia p un primo di L bit, tale che valga l'ipotesi del logaritmo discreto $\mathbb{Z}_p$, con $L \equiv 0 \pmod{64}$ e tale che $512 \leq L \leq 1024$ e preso q un primo di 160-bit che divide p-1.*
Sia $\alpha \in \mathbb{Z}_p^*$ la q-esima radice di 1 modulo p.
Definiamo

$$k = \{(p, q, \alpha, a, \beta) : \beta \equiv \alpha^a \mod p\}$$

con $0 \leq a \leq q - 1$.
I valori p,q,α e β costituiscono la chiave pubblica ed a la chiave privata; cioè: $pk = (p, q, \alpha, \beta)$ e $sk = (a)$.
Per $K = (p, q, \alpha, a, \beta)$ e per un numero random (segreto) k
tale che $1 \leq k \leq q - 1$,
si definisce

$$\boldsymbol{sig}_K(x, k) = (\gamma, \delta)$$

dove

$$\gamma = (\alpha^k \pmod p) \pmod q$$

e

$$\delta = (SHA - 1(x) + a\gamma)k^{-1} \pmod q$$

(Se $\gamma = 0$ o $\delta = 0$ deve essere scelto un nuovo valore random k)
Per $x \in \{0, 1\}^*$ e $\gamma, \delta \in \mathbb{Z}_q^*$, si ottiene che:

$$e_1 = SHA - 1(x)\delta^{-1} \pmod q$$

$$e_2 = \gamma\delta^{-1} \pmod q$$

$$\boldsymbol{ver}_K(x, (\gamma, \delta)) = true \iff (\alpha^{e_1}\beta^{e_2} \pmod p) \pmod q = \gamma$$

Osservazione 2.4.1. Con il termine **SHA** (acronimo di Secure Hash Algorithm) si indica una famiglia di cinque diverse funzioni di hash, tali che, da un messaggio di lunghezza qualunque si generi un *message digest* di lunghezza fissa.
SHA-1 produce un *digest* di 160 bit da un messaggio di lunghezza massima di $2^{64} - 1$ bit.

Passiamo ora ad analizzare l'**Elliptic Curve Digital Signature Algorithm**
Siano due punti A e B della curva ellittica E definita sul campo $\mathbb{Z}_p$ con p primo.
Il logaritmo discreto $m = \log_A B$ è la chiave privata (questa relazione è analoga a quella del sistema DSA $\beta = \alpha^a \pmod p$ con a chiave privata).

Crittosistema 2.4.2 (Elliptic Curve Digital Signature Algorithm)**.** *Sia p un primo »1 e sia E una curva ellittica definita su un campo $\mathbb{Z}_p$. Sia A un punto di E avente ordine primo q, tale che su <A> valga l'ipotesi del Logaritmo Discreto.*
Si definisce

$$k = \{(p, q, E, A, m, B) : B = mA\}$$

con $0 \leq m \leq q - 1$.
I valori p,q,E,A e B costituiscono la chiave pubblica ed m la chiave privata; cioè: pk=(p,q,E,A,B) e sk=(m).
Per K=(p,q,E,A,m,B) e per un numero random (segreto) k
tale che $1 \leq k \leq q - 1$,
si definisce

$$\boldsymbol{sig}_K(x, k) = (r, s)$$

con

$$kA = (u, v)$$

$$r = u \pmod q$$

e

$$s = k^{-1}(SHA - 1(x) + mr) \pmod q$$

(se r=0 o s=0, deve essere scelto un nuovo valore random k)
Per $x \in \{0,1\}^*$ *e* $r, s \in \mathbb{Z}_q^*$, *si ottiene che:*

$$w = s^{-1} \pmod{q}$$

$$i = wSHA - 1(x) \pmod{q}$$

$$j = wr \pmod{q}$$

$$(u, v) = iA + jB$$

$$\boldsymbol{ver}_K(x, (r, s)) = true \iff u \pmod{q} = r$$

Si riporta ora un esempio per illustrare il funzionamento del crittosistema ECDSA:

Esempio 2.4.1. Si utilizzi la curva ellittica $y^2 = x^3 + x + 6$ definita sul campo $\mathbb{Z}_{11}$, dunque $p = 11$.
Sia ora $q = 13$, $A = (2, 7)$, $m = 7$ e $B = (7, 2)$.
Supponiamo di avere un messaggio x che con SHA-1(X)=4, e Alice vuole firmare il messaggio x usando un valore random $k = 3$.
Calcoliamo

$$kA = 3(2, 7) = (u, v)$$

$$r = u \pmod{q} = u \pmod{13} = 8$$

e

$$s = 3^{-1}(4 + 7 \times 8) \pmod{13} = 7$$

Dunque (8,7) è la firma.
Bob verifica la firma eseguendo i seguenti calcoli:

$$w = 7^{-1} \bmod 13 = 2$$

$$i = 2 \times 4 \pmod{13} = 8$$

$$j = 2 \times 8 \pmod{13} = 3$$

$$(u, v) = 8A + 3B = (8, 3)$$

e

$$u \pmod{13} = 8 = r$$

Dunque la firma è verificata.

Capitolo 3

Blockchain e Mining

3.1 La Blockchain

La blockchain è una famiglia di tecnologie in cui il registro è strutturato come una catena di blocchi contenenti le transazioni e la cui validazione è affidata ad un meccanismo di consenso, distribuito su tutti i nodi.
Le principali caratteristiche di questa tipologia di tecnologie sono l'immutabilità del registro, la trasparenza, la tracciabilità delle transazioni e la sicurezza basata su tecniche crittografiche.
La blockchain può essere memorizzata come file o in un semplice database, contente una lista di blocchi di transazioni collegati "indietro", ognuno riferito al blocco precedente; essa è spesso visualizzata come una pila verticale, con blocchi sovrapposti uno sull'altro, rendendo necessario l'uso di termini come "altezza" (height) per riferirsi alla distanza dal primo blocco, e "top" per riferirsi all'ultimo blocco aggiunto.
Ogni blocco è identificato da un hash generato dall'algoritmo SHA256 applicato all'*header del blocco*, inoltre referenzia quello precedente, chiamato *parent block*, attraverso il campo *previous block hash* situato nel block header.
La sequenza di hash che collega ogni blocco al proprio parent crea una catena che si collega, blocco per blocco, fino al primo creato, il cosiddetto ***genesis***

block.
Anche se un blocco ha un solo genitore, quest'ultimo può avere più di un figlio: ogni "figlio" si riferisce allo stesso "padre" e contiene lo stesso hash nel campo previous block hash. Questo fenomeno è chiamato "**fork**"(biforcazione) della blockchain: una situazione temporanea che si verifica quando blocchi diversi vengono minati quasi contemporaneamente; questa biforcazione viene risolta ed un solo blocco figlio diventa parte della blockchain.
Il campo "previous block hash" è nel block header e quindi influenza l'hash del blocco attuale; si osserva così che l'identità del blocco figlio cambia se viene modificato il blocco padre e ciò rende necessario un cambio del puntatore "previous block hash", modificando l'identità del figlio e dunque quella del blocco "nipote". Questo assicura che, se un blocco ha tante generazioni di blocchi successive, non può essere forzato senza ricalcolare i blocchi successivi, situazione che richiede un'enorme potenza computazionale: l'esistenza di una lunga catena di blocchi fa sì che la storia più profonda della blockchain sia immutabile.
Questo è uno degli elementi chiave della sicurezza in bitcoin.

3.1.1 Il Genesis Block

Il 3 Gennaio 2009, Satoshi Nakamoto, nome con cui è stato identificato uno sviluppatore anonimo o un gruppo di sviluppatori anonimi, fece la storia quando rilasciò il Genesis Block, il primo blocco contenente 50 bitcoin.
Diversamente da quanto accade per gli altri blocchi, Nakamoto lasciò un messaggio nel codice del blocco:
"The Times 03/Jan/2009 Chancellor on brink of second bailout for banks"
("03/Gen/2009 Il cancelliere sull'orlo del secondo salvataggio delle banche").
Satoshi aveva preso spunto per il suo messaggio dal titolo di un articolo del Times uscito in quello stesso giorno che faceva riferimento al salvataggio delle banche voluto dal governo britannico. Anche se Nakamoto non spiegò mai il significato del messaggio, molti hanno letto in quelle parole il motivo che lo

avrebbe spinto a creare il Bitcoin: tagliare fuori le banche, considerate corrotte ed inaffidabili, scegliendo di creare una valuta dipendente dalle persone e con un controllo decentralizzato.
Esso è l'antenato comune a tutti i blocchi della blockchain a cui ogni altro blocco può far risalire le proprie origini dato che ognuno può essere ricondotto al precedente. Ogni nodo inizia sempre con una blockchain di almeno un blocco, perché il genesis block è staticamente codificato all'interno del client software bitcoin, in modo che non possa essere alterato; ogni nodo conosce sempre l'hash e la struttura del genesis block, pertanto ogni nodo ha il punto d'inizio della blockchain, una base sicura per costruirne una affidabile.

3.1.2 La Struttura di un Blocco

Un blocco è un contenitore di dati che aggrega più transazioni per permettere poi la loro pubblicazione nella blockchain.
Ogni blocco è costituito da un'intestazione contenente metadati e da un elenco di transazioni che ne costituiscono la parte più consistente. L'intestazione del blocco è di 80 byte, una transazione media è di circa 250 byte e un blocco medio contiene più di 500 transazioni: di conseguenza, un blocco completo, cioè contenente tutte le transazioni, risulta esser 1000 volte più grande della sua intestazione.

DIMENSIONE	CAMPO	DESCRIZIONE
4 bytes	Dimensione del blocco (Block size)	Dimensione del blocco, in byte
80 bytes	Header del blocco (Block header)	Diversi campi del Block Header
1-9 bytes	Contatore di transazione (transaction counter)	Numero di transazioni nel blocco
Variable	Transazioni	Le transazioni registrate nel blocco

L'intestazione del blocco (Block Header) consiste in tre gruppi di metadata: nel primo c'è un riferimento all'hash del precedente blocco; il secondo riguarda il mining e il terzo è costituito dal merkle root, una struttura dati usata per riassumere efficientemente tutte le transazioni del blocco.

DIMENSIONE	CAMPO	DESCRIZIONE
4 bytes	Versione	Un numero di versione per tracciare upgrade al software e/o al protocollo
32 bytes	Hash del blocco precedente	Un riferimento all'hash del blocco precedente
32 bytes	Merkle root	Hash della radice del merkle tree delle transazioni del blocco
4 bytes	Timestamp	Tempo approssimativo di creazione del blocco
4 bytes	Target di difficoltà (difficulty target)	Il target di difficoltà dell'algoritmo di proof-of-work per questo blocco
4 bytes	Nonce	Un contatore utilizzato per l'algoritmo di proof-of-work

3.1.3 Identificatori del blocco: Hash del Block Header e Block Height

Il primo identificatore del blocco è il suo hash crittografico, un'impronta digitale realizzata applicando due volte l'algoritmo SHA256 all'intestazione del blocco: il valore risultante è una stringa di 32 byte, chiamato ***Block Hash***, più precisamente è l'hash dell'intestazione del blocco che identifica in modo univoco e senza ambiguità un blocco.
Si osservi che il Block Hash non è inserito nella struttura dati del blocco, ma può essere derivato da ogni nodo della blockchain che lo riceve; inoltre questo

viene memorizzato in un separato database come una parte dei metadata dei blocchi, per facilitare l'indicizzazione ed il recupero di essi.
Un secondo modo per identificare un blocco è attraverso la sua posizione nella blockchain, detta ***Block Height***.
Il primo blocco mai creato ha altezza 0 (zero).
Un blocco può essere quindi identificato in due modi: secondo l'hash del blocco oppure secondo la sua altezza.
Ogni Blocco successivo viene aggiunto in cima al blocco precedente, assumendo quindi la posizione più alta nella blockchain.
A differenza del Block Hash, il Block Height non è un identificatore unico; nonostante un singolo blocco abbia una determinata ed immutabile altezza, quest'ultima non risulta essere un identificatore univoco dal momento che due o più blocchi possono infatti competere per una stessa posizione nella blockchain.

3.2 Il Mining

La parola "mining" evoca in noi l'estrazione di metalli, concentrando l'attenzione sulla sua ricompensa non sul procedimento; anche se è incentivato dal compenso, il mining è il processo alla base del sistema di Consenso Decentralizzato -e dunque senza il bisogno di un'autorità centrale- attraverso il quale le transazioni vengono convalidate ed inserite nel *ledger globale*, mantenendo inoltre questo meccanismo e la rete sicuri. Il mining è dunque l'invenzione che rende Bitcoin il meccanismo di sicurezza che è alla base del contante digitale *peer-to-peer*.
Un nuovo blocco viene minato in media ogni 10 minuti, grazie ad un sistema di retargeting di difficoltà, e le transazioni inserite in esso, aggiunte dunque alla blockchain, sono considerate "confermate", permettendo così ai nuovi proprietari di spendere i bitcoin ricevuti.
I miners ricevono due tipi di ricompense per il loro lavoro: i bitcoin creati ad ogni nuovo blocco e le transaction fees in esso incluse. Per ottenere queste ricompense, i miners devono competere per risolvere un difficile problema computazionale basato su un algoritmo di hashing; la soluzione al problema, chiamato *proof-of-work*, è inclusa nel nuovo blocco ed agisce come prova che il miner ha impiegato uno sforzo computazionale per arrivare alla soluzione. Vincendo questa gara, si ottiene la ricompensa ed il diritto di registrare le transazioni nella blockchain.
Il processo è detto "mining" perché la ricompensa è progettata per ottenere rendimenti decrescenti con il tempo, proprio come l'estrazione di metalli preziosi: l'offerta di moneta di bitcoin viene creata attraverso il mining ad ogni transazione. La quantità massima di Bitcoin che un minatore può ottenere come ricompensa diminuisce approssimativamente ogni 4 anni (precisamente ogni 210000 blocchi): si è iniziato con 50 bitcoin per blocco nel gennaio del 2009, si è dimezzato a 25 bitcoin nel novembre 2012 e si è arrivato a 12,5 bitcoin nel 2016. Sulla base di questo schema, i premi del mining di bitcoin diminuiranno esponenzialmente fino a circa l'anno 2140, quando tutti i 21 milioni di bitcoin saranno stati emessi.

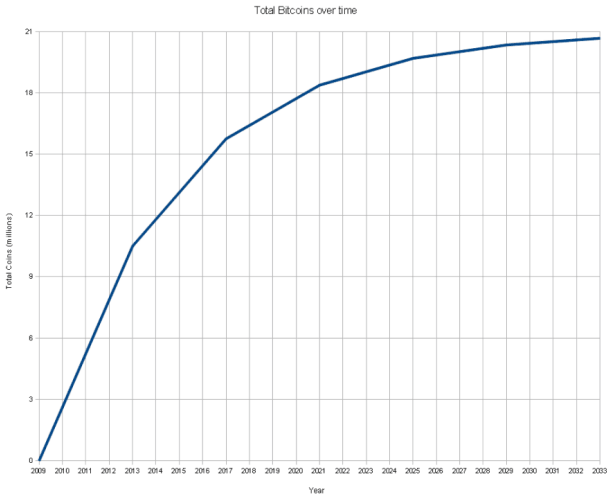

Figura 3.1: Riserva monetaria di bitcoin nel tempo basata su un ritmo di emissione decrescente

Il numero massimo di dimezzamenti consentito è 64: superato questo, il sistema impone una ricompensa zero, ottenendo come ricompensa solo le transaction fees. Per facilitare il calcolo il dimezzamento viene eseguito sulla quantità di satoshi, ovvero $50 \times 10^8 satoshi = 5$ miliardi di satoshi.

3.2.1 Consenso Decentralizzato

Tutti i sistemi di pagamento tradizionale dipendono da un modello di fiducia basato su un'autorità centrale, ma in Bitcoin questo non avviene; al contrario, infatti, ogni full node ha una propria copia del ledger a cui può affidarsi usandolo come registro autoritativo, cosicché la blockchain non viene creata da un'autorità centrale, bensì da ogni nodo della rete.
La grande invenzione di Satoshi Nakamoto è il meccanismo decentralizzato di **consenso emergente**, così definito perché il consenso non è raggiunto esplicitamente, con un'elezione o in un momento stabilito, ma viene creato dall'interazione di migliaia di nodi che seguono semplici regole.
Ogni nodo, in base alle sue proprietà, collabora al mantenimento della decen-

tralizzazione del consenso, mentre i nodi di mining assemblano le transazioni in modo indipendente dagli altri, proprio come ogni full node, verifica indipendentemente le transazioni basandosi su una serie di criteri ben definiti. Anche i nodi più semplici partecipano a questo processo, essi verificano i nuovi blocchi minati assemblandoli in una catena e selezionano quella con più calcolo cumulativo, ossia la più lunga, nel caso di biforcazioni.

3.2.2 Nodi di mining e formazione di un nuovo Blocco

Alcuni nodi della rete bitcoin sono nodi specializzati definiti **miners**: dopo aver assemblato un nuovo blocco, essi competono per crearne di nuovi eseguendo un hardware specializzato per la risoluzione di una "proof-of-work". Alcuni nodi di mining sono completi, cioè mantengono una copia della blockchain, mentre altri collaborano in "mining pool", aperti a tutti i miner, che collaborano attraverso protocolli specifici.
I minatori che partecipano ad un pool dividono il lavoro di ricerca di una soluzione per un blocco candidato, guadagnando un compenso in base al loro contributo di calcolo: quando qualcuno del pool riesce ad estrarre il blocco, il premio viene versato al pool e poi condiviso con tutti i miners in modo proporzionale al loro sforzo computazionale.

Una transazione, dopo essere stata convalidata, viene aggiunta ad una *transaction pool*, insieme ad altre per essere poi inserite nel blocco. I nodi di mining generano con queste transazioni un **candidate block**, ovvero un blocco candidato ad essere inserito nella blockchain in seguito alla competizione.
La prima transazione ad essere inserita nel blocco è detta ***coinbase transaction***, costruita dal nodo di mining e costituisce la ricompensa per l'estrazione del blocco; a differenza delle normali transazioni, questa non ha UTXO in input, ma crea bitcoin dal nulla ed ha un output, pagabile all'indirizzo bitcoin del minatore stesso.
Per costruire questa transazione, il miner calcola per prima cosa le fees

ricavate dalle transazioni nel blocco mediante la formula

$$FeesTotali = Somma(Input) - Somma(Output)$$

ed in seguito calcola la corretta ricompensa per il nuovo blocco in base all'altezza del blocco da minare.
Nelle coinbase transaction il campo dello script di sblocco è sostituito da dati della transazione compresi tra 2 e 100 byte. Questo campo è costituito solitamente da dati arbitrari che possono essere usati dai minatori in qualsiasi modo essi vogliano.
Ad esempio, nel genesis block Satoshi Nakamoto inserì il famoso testo proprio nei dati coinbase, spazio che ora invece viene utilizzato dai minatori per includere valori che identificano la mining pool. I primi campi di questa transazione sono però fissi, dal momento che devono contenere l'indice di altezza del blocco.

Per la creazione del *Block Header*, il nodo di mining deve riempire sei campi, in modo da rispettare lo schema dei blocchi della blockchain: inserirà la versione che descrive la struttura del blocco e aggiungerà il "Previuous Block Hash" selezionando così il blocco genitore; poi creerà il merkle tree in modo tale da riassumere, grazie al Merkle root, tutte le transazioni del blocco e aggiungerà quindi un timestamp ed un obiettivo di difficoltà. Il campo finale è il nonce che è inizializzato a zero.
Con tutti i campi riempiti, il Block Header è completo ed il nodo può iniziare così il processo di estrazione.

3.2.3 Il Mining di un blocco e l'algoritmo di Proof-Of-Work

L'obiettivo del Mining è quello di trovare un valore per il *nonce* che generi un hash di intestazione del blocco, inferiore all'obiettivo di difficoltà stabilito. Il nodo di mining dovrà testare dunque un'enorme quantità di valori nonce prima che venga trovato un nonce che soddisfi i requisiti.

La funzione hash utilizzata nel processo di mining bitcoin è SHA256, algoritmo che per un input di lunghezza qualunque genera un output sempre di dimensione di 256 bit.
Dunque il mining risulta essere la ripetizione dell'hashing dell'header del blocco, al quale però, ad ogni iterazione, viene modificato un parametro, fino a quando l'output soddisfa una determinata condizione.

Le caratteristiche delle funzioni di hash implicano che l'unico modo per trovare il risultato è la ripetizione del processo fino all'ottenimento del valore richiesto e che, modificando anche un singolo carattere dell'input, gli output ottenuti risultano essere completamente differenti.
Se si applica la funzione HASH256 alla frase "I am Satoshi Nakamoto" e successivamente la si applica nuovamente aggiungendo un numero si ottiene:
· I am Satoshi Nakamoto ⇒
5d7c7ba21cbbcd75d14800b100252d5b428e5b1213d27c385bc141ca6b47989e
· I am Satoshi Nakamoto0 ⇒
a80a81401765c8eddee25df36728d732acb6d135bcdee6c2f87a3784279cfaed
· I am Satoshi Nakamoto1 ⇒
f7bc9a6304a4647bb41241a677b5345fe3cd30db882c8281cf24fbb7645b6240
...
· I am Satoshi Nakamoto13 ⇒
0ebc56d59a34f5082aaef3d66b37a661696c2b618e62432727216ba9531041a5

Come si vede si ottengono valori sempre diversi.

Per creare una challenge, si imposta un obiettivo arbitrario che l'output deve rispettare: nel caso appena analizzato potrebbe essere quello di utilizzare un input che produce un hash esadecimale che inizi per 0, qui sono stati necessari ben 13 tentativi ed il **nonce** vincente è 13.
In termini matematici, impostare un certo target si traduce nel trovare un output inferiore ad un certo valore, o soglia, chiamato **target**.

Osservazione 3.2.1. La Proof-of-work deve produrre un hash inferiore ad un dato target: più la soglia è bassa, maggiore è lo sforzo computazionale necessario, di conseguenza si può affermare che obiettivo e difficoltà sono inversamente correlati.

Ovviamente l'aumento della difficoltà di un bit provoca una crescita esponenziale del tempo necessario per trovare una soluzione: nello spazio di 256 bit, ogni volta che si limita un bit a zero, si riduce lo spazio di ricerca della metà; più sono gli zeri richiesti all'inizio dell'hash, più piccolo è l'intervallo accettabile di output.
Il sistema Bitcoin regola la difficoltà in modo che la soluzione al problema sia trovata in media ogni 10 minuti.
Trovata la soluzione alla Proof-of-work, il nodo vincitore trasmette il blocco a tutti i nodi della rete che eseguono un controllo indipendente.
Un meccanismo alla base del consenso generalizzato è appunto la convalida indipendente di ogni nuovo blocco da parte di ogni nodo della rete, che assicura l'onestà dei minatori durante il processo di estrazione.
Se un minatore non dovesse agire in modo onesto, il blocco da lui minato con grande sforzo computazionale non verrebbe accettato dal resto della rete, comportando così la mancata compensazione del costo dell'elettricità utilizzata.

3.2.4 Il Retargeting della Difficoltà

All'interno del block header si trova un campo dedicato al target di difficoltà dell'algoritmo di proof-of-work che influenza il tempo necessario per trovare una soluzione.
In media i blocchi vengono generati ogni 10 minuti e questa velocità deve rimanere costante, non solo nel breve periodo, ma addirittura in un lasso di tempo di molti decenni.
Si prevede che la potenza dei computer continuerà ad aumentare ad un ritmo rapido ed il numero di partecipanti nel settore del mining, come anche la potenza dei calcolatori utilizzati, cambierà costantemente ed è per questo

che il retargeting di difficoltà si verifica automaticamente e su ogni nodo in modo indipendente.
Ogni 2016 blocchi, tutti i nodi modificano la difficoltà della proof-of-work misurando il tempo impiegato per minare gli ultimi 2016 blocchi e lo confrontano con il tempo previsto di 20160 minuti ($2016 \times 10min = 20160$). Viene dunque calcolato il rapporto tra periodo effettivo e periodo teorico e viene applicata una correzione alla difficoltà secondo la formula:

$$\text{New Target} = \text{Old Target} \times \frac{\text{Tempo attuale per minare 2016 blocchi}}{\text{Stima teorica} = 20160 \text{ minuti}}$$

La regolazione del retargeting, però, non può essere maggiore di un fattore 4 per ciclo: se il coefficiente moltiplicativo sarà maggiore di tale quantità, sarà regolata al massimo, ma non di più e se dunque si avrà un grande squilibrio, saranno necessari più cicli da 2016 blocchi per bilanciare il sistema; questo per evitare che il sistema diventi eccessivamente instabile.
Si osservi che la difficoltà non dipende dal numero di transazioni o dal valore di queste all'interno dei blocchi: infatti la potenza di hashing traduce le forze computazionali che partecipano all'estrazione dei blocchi e per questo è strettamente collegata al costo dell'elettricità e al tasso di cambio del bitcoin rispetto alle valute convenzionali.

3.2.5 Assemblamento della Blockchain e Fork

Il passo finale del meccanismo di consenso decentralizzato di bitcoin è l'assemblaggio di catene di blocchi e la selezione di quella con la maggiore prova di lavoro. Dopo che un nodo convalida un blocco, tenterà di assemblare una catena collegandolo alla blockchain esistente.
I nodi collegano tre tipi di blocchi: quelli che possono essere connessi alla blockchain principale, quelli che formano rami partendo dalla blockchain principale ed infine blocchi che non hanno un genitore, detti "orfani".
La catena principale sarà quella a cui è associata la prova di lavoro maggiore e risulta essere, nella maggior parte dei casi, quella con il più alto numero di blocchi. Da questa partono, in determinate situazioni, dei rami con blocchi

"fratelli" a quelli della catena principale: si formano così i **fork** della blockchain.

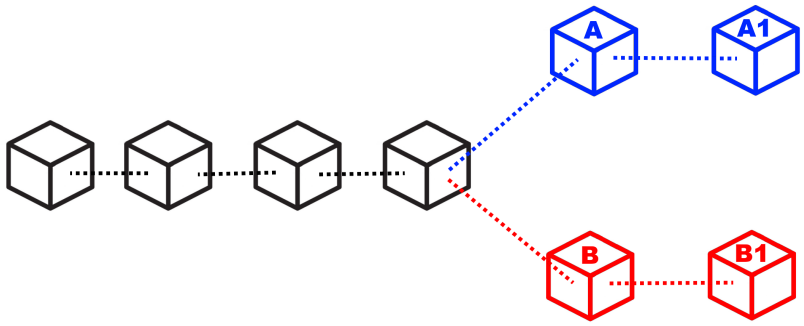

Figura 3.2: Un fork della Blockchain

Poiché la blockchain è una struttura decentralizzata, le diverse copie non sono sempre identiche. I blocchi potrebbero arrivare ai vari nodi in momenti diversi ed è per questo che può capitare che questi ultimi abbiano prospettive differenti della blockchain. Per risolvere questo problema, ogni nodo andrà ad estendere la catena con maggiore proof-of-work: sommando la difficoltà registrata ad ogni blocco di una chain è possibile calcolare la quantità totale di proof-of-work che è stata spesa per crearla.
I fork sono passaggi temporanei tra varie versioni della blockchain, che sono risolti mano a mano che più blocchi sono aggiunti ad uno dei rami.

Osservazione 3.2.2. I fork della blockchain si verificano naturalmente a causa dei ritardi di trasmissione nella rete globale, che variano in modo casuale.

Ogni nodo ha la propria prospettiva della blockchain globale, poiché questo riceve i blocchi dai propri vicini e aggiorna la propria versione della blockchain in base anche alla sua posizione nel network.

Un fork si verifica ogni qual volta due minatori risolvono l'algoritmo di proof-of-work quasi simultaneamente (generando due blocchi α e β). Quando

i miners trovano la soluzione, inviano i blocchi ai loro vicini iniziando così la propagazione nella rete. Se i nodi ricevono un altro blocco candidato che estende lo stesso genitore, lo collegano ad una catena secondaria: come conseguenza si ha che non tutti i nodi vedono con lo stesso ordine i blocchi candidati, con la successiva formazione di un fork. Entrambi i blocchi inviati sono validi ed estendono lo stesso genitore, contenendo molto probabilmente anche le stesse transazioni; è per questo motivo che entrambe le visioni della blockchain ($\alpha \implies$ **A** e $\beta \implies$ **B**) sono valide. Solo con mining successivi si capirà quale delle due catene (**A** o **B**) verrà aggiunta definitivamente al libro mastro.
Subito dopo l'estrazione del blocco, i nodi di mining cominciano a costruire un nuovo candidato con uno dei due blocchi sopra analizzati come genitore, in base alla loro visione della catena principale. Dopo circa 10 minuti verrà trovata una soluzione al problema computazionale successivo ed inizia così la propagazione del nuovo blocco.
A questo punto il fork verrà risolto in quanto solo uno dei due blocchi sarà il genitore di quello appena minato (ad esempio il blocco α).

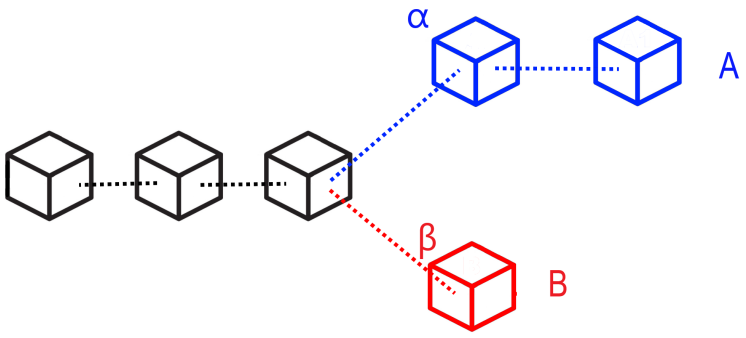

Figura 3.3: Estensione di una delle due catene e conseguente risoluzione del fork

Tutti i miners che stavano lavorando per estendere la catena **B** interrompono il loro lavoro, poiché il loro blocco candidato è diventato un "orfano".
A questo punto l'intera rete riconfigura una singola blockchain.

Le transazioni inserite in β e non in α vengono reinserite nella "transaction pool" per essere poi incluse nei blocchi successivi.

La possibilità di estensione dei fork di due blocchi è teoricamente possibile, tuttavia la possibilità che ciò accada è molto bassa: infatti, mentre un fork potrebbe verificarsi ogni settimana, uno a due blocchi è estremamente raro.

Conclusioni

Le Blockchain rappresentano una modalità particolarmente trasparente e decentralizzata per la registrazione di transazioni.
Poiché le transazioni hanno fees molto basse (intorno ai 0,1 €), ma offrono una registrazione sicura e permanente nel ledger, la Blockchain viene utilizzata anche per finalità non finanziarie, dalla votazione all'approvazione di brevetti.
Certo è vero che la natura decentralizzata delle applicazioni porterebbe ad un'autoregolazione che entra in conflitto con la natura centralizzata delle istituzioni, generando così un loro significativo cambiamento; per ogni transazione che usa un libro mastro distribuito, gli intermediari e i mediatori perdono potere e reddito.
Ad esempio, sarebbe possibile regolamentare ed unificare il sistema dei brevetti in modo tale da difendere da violazioni illegali quelle aziende che, non essendo in grado di sostenere il prezzo della protezione brevettuale, decidono di lanciare sul mercato le innovazioni senza proteggerle.
Interessante è anche il concetto di Smart Contract, definito nel 1994 come un "un protocollo di transazione computerizzato che esegue i termini di un contratto". La blockchain può avere un ruolo attivo e autonomo nella registrazione delle transazioni, permettendo così la loro verifica ed esecuzione solo in seguito al verificarsi di determinate condizioni, fornendo così una "garanzia di esecuzione".

Numerosi sono gli Stati che utilizzano la Blockchain per gestire procedu-

re tecniche ed amministrative, dai registri catastali (come Ghana, Kenya e Nigeria) ai pagamenti assistenziali delle pensioni (come il Regno Unito).

Lo Stato italiano non ha ancora deciso di utilizzare questa tecnologia innovativa, ma la sua introduzione nelle pubbliche amministrazioni potrebbe permettere l'ottimizzazione delle procedure, con una conseguente riduzione dei costi e soprattutto l'ottenimento di un sistema più affidabile.
È un esempio il comune abruzzese Miglianico, il quale, aderendo al principio europeo "chi inquina paga", ha regolarizzato questo concetto mediante l'utilizzo della Blockchain: i mastelli possiedono un tag RFID che vengono letti durante le operazioni di raccolta dei rifiuti. Le transazioni relative alla raccolta rifiuti sono registrate sul sistema inviolabile in modo tale da permettere al cittadino di avere un database preciso sul consumo e al comune di verificare eventuali frodi.

Bibliografia

[1] Rafael Pass, Abhi Shelat. *A Course in Cryptography*, URL: //www.cs.cornell.edu/courses/cs4830/2010fa/lecnotes.pdf.

[2] Andreas M. Antonopoulos, *Mastering Bitcoin. Unlocking digital cryptocurrencies*, Sebastopol, O'Reilly Media, Inc., 2014, ISBN: 978-1-449-37404-4.

[3] Nakamoto Satoshi, *Bitcoin: A Peer-To-Peer Electronic Cash System*

[4] Douglas R. Stinson, *Cryptograrhy. Teory and Practice*, Boca Raton, Chapman & Hall/CRC, 2006.

[5] Andrea Corbellini, *Elliptic Curve Cryptography: a gentle introduction.*

[6] Philip Boucher, *Come la tecnologia blockchain può cambiarci la vita*, Bruxelles, Servizio di ricerca del Parlamento europeo, 2017.