Introduzione alla blockchain

Y2Y: BLOCKCHAIN E SMART CONTRACT

Organizzatori: Massimiliano Sala, Andrea Gangemi e Christopher Spennato

Speakers: Andrea Flamini ed Enrico Guglielmino

12 e 19 Ottobre, 2023

Sommario

Crittografia

Introduzione alla crittografia

La crittografia si occupa delle tecniche e degli algoritmi per rendere un messaggio incomprensibile a persone non autorizzate a leggerlo.

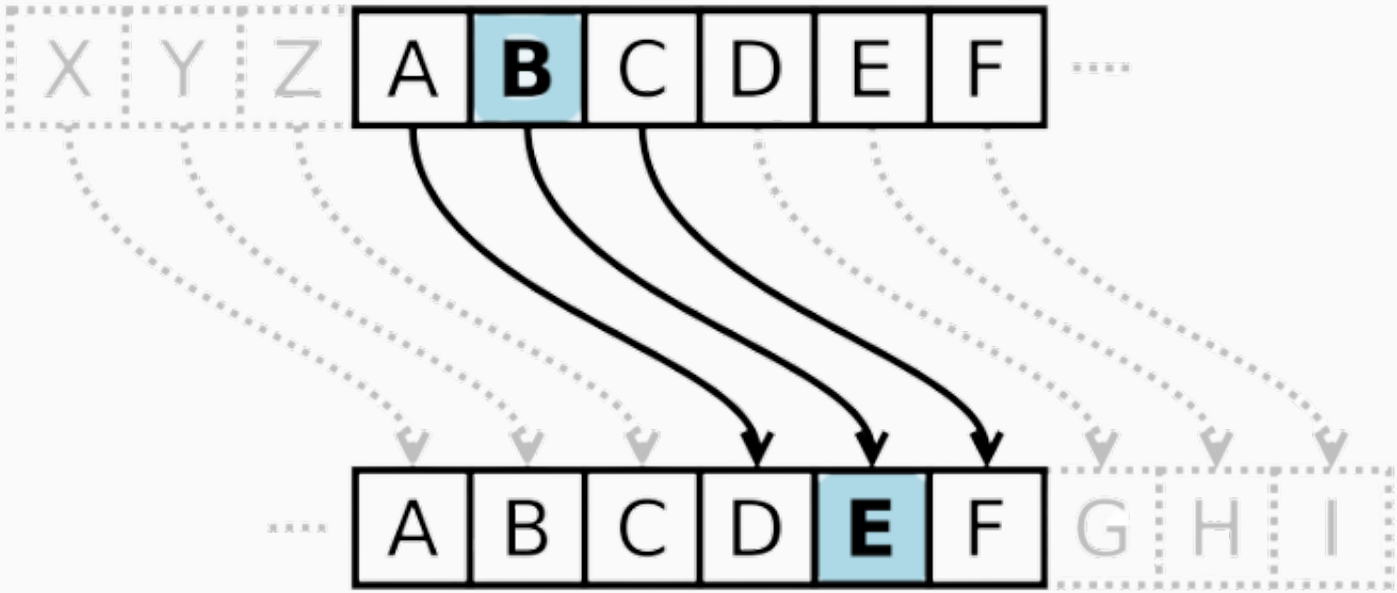
Il modo classico per risolvere questo problema è costruire un algoritmo con un parametro mantenuto segreto. Coloro che conoscono questo parametro, chiamato **chiave**, possono applicare l'algoritmo trasformando il testo in chiaro in testo cifrato e viceversa.

Si parla in questo caso di **crittografia simmetrica**, in quanto la decifratura utilizza la stessa chiave segreta della cifratura e lo stesso algoritmo eseguito al contrario.

Crittografia simmetrica

Esempio: Cifrario di Cesare I

Un modo molto semplice per cifrare un messaggio, che sembra risalire a Giulio Cesare, è quello di "shiftare" tutte le lettere dell'alfabeto.



Esempio: Cifrario di Cesare II

Considerando le 26 lettere dell'alfabeto inglese, ossia ABCDEFGHIJKLMNOPQRSTUVWXYZ e spostandole in avanti di 3 posizioni (cioè con chiave di cifratura uguale a 3), il testo in chiaro **"DE CIFRIS"** viene crittografato in **"GH FLIULV"**.

Se, d'altra parte, si cambia la chiave e si sceglie il valore 5, abbiamo che lo stesso testo in chiaro **"DE CIFRIS"** viene cifrato in **"IJ HNKWNX"**.

Esempio: Cifrario di Cesare III

Un grave difetto del Cifrario di Cesare è che le possibili chiavi di cifratura sono soltanto 25, quindi chi conosce il cifrario può provarle tutte e decifrare facilmente (attacco *brute force*).

Infatti, il cifrario di Cesare, come molti dei cifrari più antichi, basa la sua sicurezza sulla segretezza dell'algoritmo stesso.

Costruendo invece un sistema con un numero eleato di chiavi, si può ottenere un protocollo sicuro anche se l'algoritmo è noto e viene tenuta segreta solo la chiave.

Sistemi di Crittografia a Chiave Pubblica I

Un altro problema dei sistemi di crittografia a chiave simmetrica è che la chiave deve essere nota **soltanto** al mittente e al destinatario del messaggio.

Per superare il problema della distribuzione delle chiavi, Diffie e Hellman hanno proposto nel 1976 l'uso di un Sistema di Crittografia a Chiave Pubblica.

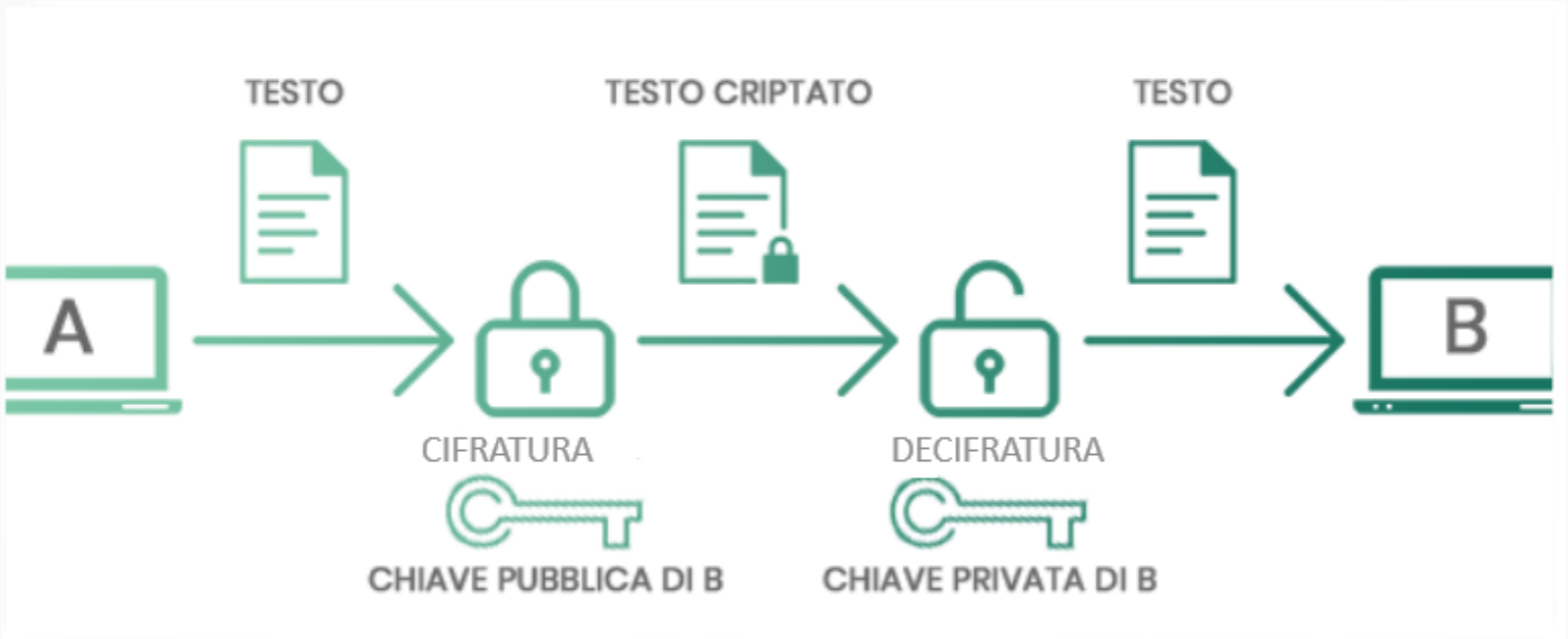
Il primo e più famoso dei sistemi a chiave pubblica è RSA (dalle iniziali dei suoi inventori Rivest, Shamir e Adleman) ed è stato pubblicato nel 1977.

Sistemi di Crittografia a Chiave Pubblica II

Nei sistemi a chiave pubblica, chiamati anche sistemi di cifratura asimmetrica, **ogni utente ha due chiavi: una pubblica e una privata**. Non c'è il problema dello scambio di chiavi, in quanto ogni utente può generarle indipendentemente.

La chiave pubblica può essere facilmente derivata a partire dalla chiave privata, mentre il contrario non è vero. Le due chiavi sono collegate da un problema matematico che si assume sia difficile da risolvere.

Sistemi di Crittografia a Chiave Pubblica III



Sistemi di Crittografia a Chiave Pubblica IV

La chiave pubblica di un destinatario viene utilizzata da qualunque mittente voglia cifrare un documento a lui indirizzato, mentre la chiave privata viene utilizzata dal destinatario per effettuare la decifratura. **È l'unico in grado di decifrare il documento**!

L'introduzione della crittografia a chiave pubblica non ha reso obsoleta la crittografia simmetrica, a causa della lunghezza delle chiavi superiore e del tempo medio più lungo richiesto per cifrare/decifrare.

Crittografia a chiave pubblica e crittografia simmetrica sono infatti spesso utilizzate insieme.

ECC

La Crittografia su Curve Ellittiche (*Elliptic Curve Cryptography*, spesso abbreviata con ECC) è stata proposta indipendentemente da Neal Koblitz e Victor Miller nel 1985.

La ECC è più efficiente rispetto a tutte le alternative di crittografia a chiave pubblica, compreso RSA (ad esempio, ECC con chiavi da 256 bit è più sicura di RSA con chiavi da 4096 bit), ma è anche più complicata da comprendere.

La complessità dell'ECC ha ritardato la sua adozione, entrando nei sistemi crittografici standard solo nei primi anni 2000.

Funzioni di Hash

Un altro importante strumento in crittografia sono le **funzioni di hash**. Queste funzioni trasformano una sequenza binaria di lunghezza arbitraria in una sequenza di bit di lunghezza prefissata.
L'output di una funzione di *hash* h è generalmente chiamato **digest**.
Inoltre, godono delle seguenti proprietà:

- sono facili e veloci da calcolare;
- è computazionalmente inefficiente invertirle, cioè dato un digest z è difficile trovare un x tale che $h(x) = z$;
- è quasi impossibile trovare *collisioni*, ossia trovare due valori diversi a e b tali che $h(a) = h(b)$;
- deve valere *l'effetto valanga*: cambiare un singolo bit nell'input cambia completamente l'output della funzione di *hash*.

Per testare le funzioni di *hash* si possono usare dei tool online.

Esempio

A titolo di esempio, usiamo il *tool online* per calcolare la funzione *hash*, utilizzando SHA-256 delle parole 'roma' e 'Roma'.

$$\text{hash(roma)} =$$
$$\text{b0c27fca74fa91934900c9ffcb3dcca5b807a3c059a3b516cdd0788807b5ff49}$$

$$\text{hash(Roma)} =$$
$$\text{7e325f21b5b5f02606012cea67fa1f55e4cfb385892fefa4d081060287310dd3}$$

Uso delle funzioni di hash per la sicurezza

Se calcoliamo l'*hash* di un *file* e lo memorizziamo da qualche parte (ad esempio, su una *blockchain*), allora in futuro chiunque può calcolare nuovamente questo *hash* e verificare se è uguale a quello memorizzato.

Questo perché siccome è quasi impossibile trovare collisioni, in teoria nessuno è in grado di modificare il file e lasciare invariato l'*hash*. Attraverso l'uso delle funzioni di *hash*, è quindi possibile verificare l'immutabilità di un pezzo di informazione.

Ci sono molteplici funzioni di *hash*, ma le più utilizzate nell'ambiente delle *blockchain* sono RIPEMD160, SHA256 (SHA2) e Keccak256 (SHA3).

Firma digitale I

Una **firma digitale** serve a garantire il mittente di un messaggio o, più in generale, l'identità di un utente che esegue una determinata operazione.

I Sistemi a Chiave Pubblica hanno come vantaggio il numero ridotto di chiavi, evitando anche il problema dello scambio di chiavi. D'altra parte, poiché chiunque può inviare messaggi cifrati usando la chiave pubblica del destinatario, sorge il problema di autenticare il mittente del messaggio.

In particolare, le firme digitali garantiscono il **non ripudio**, ovvero il mittente non può negare di avere inviato un messaggio che ha effettivamente inviato.

Firma digitale II

Ogni crittosistema a chiave pubblica ha anche la propria firma digitale: per questo motivo, ci sono, ad esempio, firme digitali basate su RSA.

Ci sono anche dei *framework* che consentono la creazione di una firma digitale, senza passare da un crittosistema. La firma più famosa e utilizzata ottenuta in questo modo è il Digital Signature Algorithm (DSA), approvato dal *National Institute of Standards and Technology* (NIST) nell'agosto 1991 e oramai standard da più di 30 anni.

Esiste una variante di DSA basata sulle curve ellittiche, chiamata Elliptic Curve Digital Signature Algorithm (ECDSA).

Firma di Schnorr e firma BLS

Bitcoin ed Ethereum, le due blockchain più importanti fino ad oggi (2023), utilizzano ECDSA per garantire la sicurezza delle transazioni.

Questo tipo di firma è ancora considerato sicuro, ma non molto flessibile. In alcuni contesti, potrebbe essere utile avere firme multiple o firme soglia, consentite da ECDSA soltanto in modo limitato.

Per questo motivo, dal novembre 2021, Bitcoin sta consentendo l'uso delle firme di Schnorr, mentre Ethereum sta sperimentando da un po' di tempo la firma Boneh-Lynn-Shacham (BLS).

Introduzione alla blockchain

Introduzione alla tecnologia blockchain

La **blockchain** è una struttura dati che consiste in un elenco di blocchi collegati tra loro in modo sicuro grazie all'utilizzo di algoritmi crittografici.

Le caratteristiche che accomunano i sistemi sviluppati con le *blockchain* sono: digitalizzazione dei dati, decentralizzazione, disintermediazione, tracciabilità e trasparenza, immutabilità.

La tecnologia è nata nel 2009, dopo l'avvento di **Bitcoin**, e si è subito dimostrata una delle tecnologie più dirompenti degli ultimi anni.

Vediamo ora un po' più nel dettaglio gli eventi storici più importanti che hanno portato alla nascita di Bitcoin, insieme ad alcune delle proprietà di base delle *blockchain*.

Un po' di storia

- **1993** - **Cypherpunk Manifesto** (Eric Hughes)
- **1997** - **Hashcash** (Adam Back)
- **1997** - **L'idea degli smart contract** (Nick Szabo)
- **1998** - **B-money** (Wei Dai)
- **2004** - **Reusable Proof of work** (Hal Finney)
- **2005** - **Bit gold** (Nick Szabo)
- **2008** - **Fallimento di Lehman Brothers**
- **2008** - **Bitcoin** (Satoshi Nakamoto)
- **3 gennaio 2009** - **Blocco Genesi di Bitcoin**
- **2015** - **Ethereum** (Vitalik Buterin)
- **Oggi** - **oltre 9.000 altcoin**

1997 - Hashcash I

Hashcash è un meccanismo proposto nel marzo 1997 da Adam Back, utilizzato per limitare le email di *spam* e gli attacchi di *denial of service*.

L'idea è la seguente: per inviare un'email, ogni utente deve prima creare un "timbro" da allegare all'email stessa. Il timbro viene creato aggiungendo una stringa casuale al messaggio e calcolando il *digest* del messaggio usando una funzione di *hash* con output di lunghezza 160 bits, finché questo *digest* non inizia con 20 zeri.

Date le proprietà delle funzioni di *hash*, l'unico modo per procedere è aggiungere casualmente una stringa. La probabilità che l'*hash* inizi con 20 zeri può essere calcolata dal rapporto tra i casi favorevoli (2^{140}, il numero di stringhe a 160 bit che iniziano con 20 zeri) e i casi possibili (2^{160}).

In media, sono quindi necessari circa 2^{20} tentativi per produrre un timbro.

1997 - Hashcash II

Pertanto, per creare un timbro valido è necessario del tempo, che dipende anche dalla fortuna e dalla potenza della propria CPU, mentre è sufficiente una frazione di secondo per verificarne la correttezza: **basta infatti calcolare un singolo hash!**

Naturalmente, ci sono anche degli svantaggi:

- Coloro che gestiscono legittimamente delle *mailing list* devono inviare molte email, e rischiano di utilizzare troppo tempo per calcolare tutti i timbri.
- Non è banale stabilire il numero corretto di zeri richiesti per il timbro, anche perché varia con l'aumento della potenza di calcolo dei computer.
- Gli *spammer* possono dotarsi di **ASIC** (*Application Specific Integrated Circuit*), progettati appositamente per calcolare *hash* molto rapidamente.

1997 - L'idea degli smart contract I

Gli **smart contract** sono protocolli informatici che facilitano, verificano o eseguono automaticamente la negoziazione o l'esecuzione di un contratto.

L'idea degli *smart contract*, completamente svincolata dalla *blockchain* che doveva ancora essere teorizzata, è attribuita a Nick Szabo e risale al 1997.

Come vedremo in dettaglio nel resto del corso, oggi con il termine *smart contract* ci si riferisce a un programma eseguito sui nodi validatori di una *blockchain* e il cui risultato, che corrisponde a una modifica dello stato della *blockchain* stessa, rappresenta una transazione su cui i nodi validatori devono essere d'accordo.

Gli *smart contract* aspirano a fornire maggiore sicurezza e velocità di esecuzione rispetto ai contratti tradizionali.

1997 - L'idea degli smart contract II

L'esempio dato da Szabo come precursore degli *smart contract* nella vita reale è il **distributore automatico**: un sistema automatizzato che, ogni volta che riceve una moneta e l'indicazione di un prodotto, eroga automaticamente il prodotto desiderato.
Fondamentalmente, il distributore automatico è un contratto con il fornitore: chiunque possegga monete può partecipare a uno scambio, mediante regole definite a priori dal distributore automatico stesso.

Con *software* di questo tipo si potrebbero ridurre notevolmente le controversie, la necessità di controllo, le spese assicurative e così via, riducendo i costi e automatizzando la gestione.

1998 - B-money (Wei Dai) I

B-money è stata una delle prime proposte di criptovaluta, ideata da Wei Dai con l'obiettivo di costruire un sistema di contante elettronico anonimo e distribuito.

L'idea è stata proposta durante una discussione sulla *mailing list* dei *cypherpunk*, relative a possibili applicazioni di *Hashcash*, che era stata a sua volta pubblicata sulla stessa *mailing-list* l'anno precedente.

L'utilizzo di *Hashcash* era visto come possibile mezzo per creare del denaro elettronico in modo sicuro.

1998 - B-money (Wei Dai) II

Il protocollo di consenso di *b-money* era però più complesso e indefinito. Per esempio, i contratti potevano essere stipulati con possibile riparazione in caso di inadempienza, con una terza parte che acconsentiva a essere l'arbitro.

Wei Dai ha proposto anche un secondo protocollo dove i conti sono tenuti solo da un sottoinsieme dei partecipanti (i *server*), che devono dare una cauzione in denaro. La cauzione viene persa se il server è ritenuto disonesto.

Molte delle idee di Dai sono state riprese da Nakamoto per la creazione di Bitcoin, e in seguito da altre criptovalute.

2004 - Reusable Proof of work (Hal Finney)

Nel 2004, Finney ha ideato un protocollo che, similmente a *b-money*, crea valuta utilizzando *hashcash*, ma che offre anche la possibilità di trasferire in modo sicuro e non ripudiabile la criptovaluta da un utente all'altro.

Il sistema riceve *token* creati con *hashcash*, e in cambio crea *token* firmati RSA, che Finney chiama **Reusable Proof of Work (RPoW)**. I *token* RPoW possono quindi essere trasferiti da persona a persona e scambiati con nuovi RPoW a ogni passaggio.

Ogni *token* può essere utilizzato solo una volta ma, poiché ne dà vita a uno nuovo, è come se lo stesso gettone potesse essere consegnato da persona a persona.

2005 - Bit gold (Nick Szabo)

Szabo ha cominciato a ragionare su una moneta digitale anonima e distribuita nel 1998, ma ha pubblicato la proposta di **Bit gold** solo nel 2005.

Come i token RPoW di Finney, anche *Bit gold* crea monete utilizzando *hashcash*. Si parte da una stringa casuale pubblica (la *challenge string*) che gli utenti possono combinare con una loro personale stringa, aggiungere un *timestamp* (distribuito) e calcolare l'*hash* fino a che non risulta minore di un certo *target*.

Il primo che ottiene l'*hash* minore del target registra la *challenge string* e la *proof of work string*, con un *timestamp*, in un registro pubblico distribuito.

2008 - Fallimento di Lehman Brothers

La *Lehman Brothers Holdings Inc.*, fondata nel 1850, era una delle più grandi banche d'investimento del mondo. Considerata "troppo grande per fallire", ha dichiarato bancarotta il 15 settembre 2008 con centinaia di miliardi di debiti.

Il fallimento della *Lehman Brothers* ha aumentato i dubbi di molti sul sistema economico occidentale e ha spinto coloro che erano già su posizioni critiche, come il movimento *cypherpunk*, a cercare un'alternativa.

Non è un caso che l'idea di Bitcoin sia nata nel 2008.

2008 - Bitcoin

Un inventore anonimo, conosciuto con lo pseudonimo di **Satoshi Nakamoto**, ha pubblicato un articolo alla fine del 2008 intitolato Bitcoin: A Peer-to-Peer Electronic Cash System, in cui ha definito la creazione di una criptovaluta digitale, anonima e distribuita, gestita da una rete di utenti *peer-to-peer*.

Non è chiaro se Nakamoto sia (o fosse) una persona o un gruppo di persone, ma molti sospettano che abbia qualcosa a che fare con Nick Szabo, Hal Finney, Wei Dai e/o altri attivisti del movimento *cypherpunk*.

3 gennaio 2009 - Blocco Genesi di Bitcoin

Il 18 agosto 2008 è stato registrato il dominio **bitcoin.org** e, dopo la pubblicazione del *whitepaper*, il 3 gennaio 2009, è stato creato il blocco genesi (il primo blocco della *blockchain*) di *Bitcoin*.

L'11 gennaio 2009, Hal Finney ha annunciato su *Twitter* come funziona *Bitcoin*, e il giorno successivo, tra Satoshi Nakamoto e Hal Finney, è avvenuta la prima transazione ufficiale dal valore di 10 *bitcoin* (BTC).

Il 22 maggio 2010 è avvenuta la prima transazione del mondo reale utilizzando *bitcoin* come valuta. Un programmatore di Jacksonville, in Florida, Laszlo Hanyecz, paga 10000 *bitcoin* per due pizze.

Il 6 novembre dello stesso anno, la capitalizzazione di *Bitcoin* ammonta a un milione di dollari e il tasso di cambio della criptovaluta su MtGox (un *exchange*) raggiunge una quotazione di mezzo dollaro per *bitcoin*.

Bitcoin

Per molti anni, la valutazione di *bitcoin* è aumentata molto lentamente ed è stato solo dopo il 2017, con l'aumento significativo dell'uso, che il suo valore è iniziato a salire in modo significativo.

Valore di Bitcoin dalla sua nascita fino ad agosto 2023

2015 - Ethereum

Ethereum è stata fondata nel 2015 da Vitalik Buterin. Non è solo una nuova criptovaluta, ma rappresenta un cambiamento di paradigma nel mondo delle *blockchain*.

Ethereum è una piattaforma globale e *open source* basata su *blockchain*, su cui possono essere eseguite applicazioni decentralizzate (dApp).

Come *Bitcoin*, ha la sua criptovaluta (*ether* - ETH), ma *Ethereum* è principalmente uno strumento per gestire ciò che è essenzialmente un enorme computer globale, che non può essere spento e le cui applicazioni, una volta avviate, non possono essere bloccate.

Quando ci si riferisce a *Ethereum*, si parla anche di *blockchain* 2.0.

Oggi - oltre 9000 altcoin

Dalla nascita di *Bitcoin* nel 2009, sono nate molte altre criptovalute.

Ad oggi (agosto 2023), sul portale *Coinmarketcap* (**https://coinmarketcap.com/it/**) ne sono elencate ben 9.580.

Tra le più importanti *blockchain*, oltre a quelle già descritte, ci sono *Ripple, Dash, Monero, Algorand e Cardano*.

Blockchain

L'obiettivo principale di una **blockchain** è creare una rete tra persone che non si conoscono e, quindi, non hanno motivo di fidarsi l'una dell'altra, consentendo interazioni sicure tra di loro senza la necessità di un'autorità centrale.

Creare scambi o relazioni sicure tra estranei è un problema comune nella nostra società. In effetti, la soluzione classica è fare affidamento su un terzo di fiducia (sovrano, stato, banca, governo locale, e così via).

Caratteristiche della blockchain

Una *blockchain* può essere descritta come:

- un *database* distribuito (Distributed Ledger Technology - DLT);
- un registro non modificabile da singoli utenti o gruppi di utenti;
- una piattaforma che introduce il concetto di scarsità digitale.

Inoltre, ogni *blockchain* è associata a una **criptovaluta** o **crittovaluta**. Le più importanti (per adozione e capitalizzazione di mercato) sono il **bitcoin** e l'**ether**.

Distributed Ledger Technology I

In passato, le informazioni su qualsiasi transazione venivano semplicemente inserite in un libro ordinario, il cosiddetto **registro**. Un registro è un libro, un quaderno, un file o un volume in cui sono registrati in un ordine specifico atti e fatti di carattere principalmente contabile e legale. Inoltre, tutta la fiducia è nelle mani dell'utente che possiede il registro. Tuttavia, ciò ha i seguenti problemi:

- **Furto:** chiunque può rubare il registro, cancellando o modificando le informazioni;
- **Fattore umano:** è facile scrivere informazioni errate al posto di quelle corrette, intenzionalmente o per errore;
- **Forza maggiore:** il registro può essere distrutto per motivi naturali, come un'alluvione o un incendio.

Una possibile soluzione è creare molte copie di questo registro.

Distributed Ledger Technology II

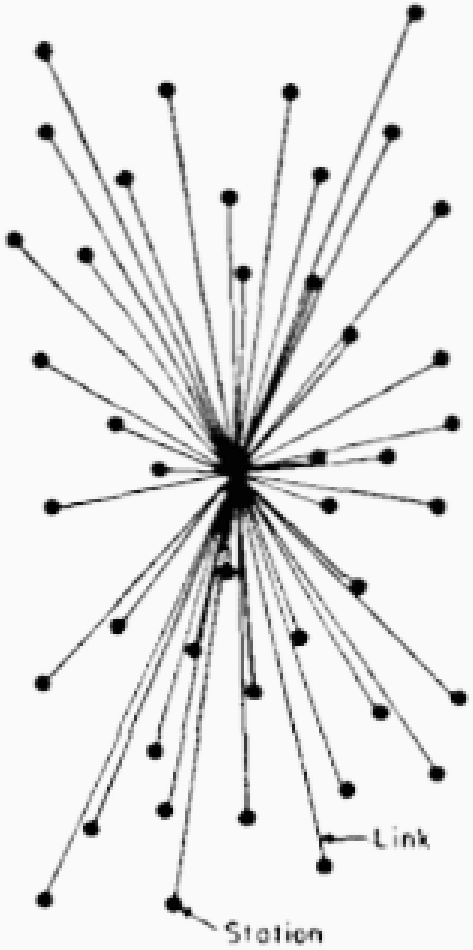
In generale, i problemi dei registri vengono risolti in modo semplice: basta fare più copie! Infatti, la DLT comporta la duplicazione dei dati da archiviare tra (molti) nodi in una rete.

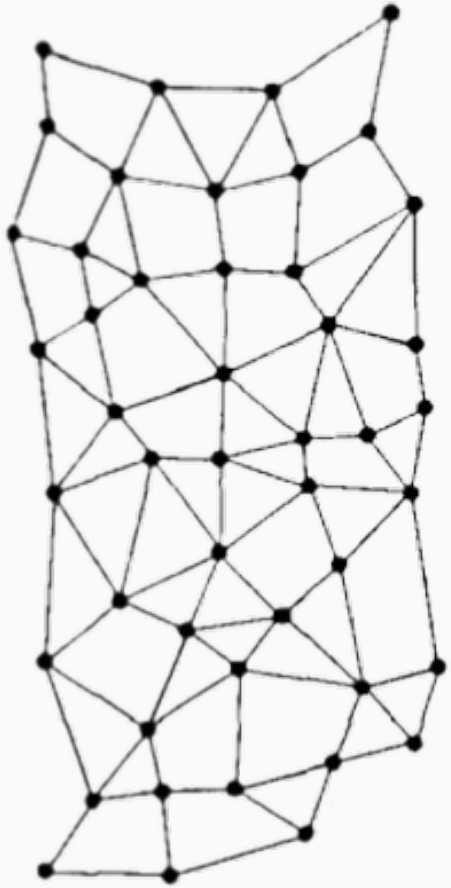
Vantaggi della DLT

- **Trasparenza e immutabilità:** a differenza di un sistema centralizzato, tutti i nodi godono degli stessi diritti sui dati. Tutte le decisioni vengono prese collettivamente. La DLT fornisce un percorso di verifica di tutte le operazioni;

- **Resistenza agli attacchi:** la DLT è un sistema più resistente agli attacchi informatici rispetto ai database centralizzati tradizionali perché è distribuita. Non c'è un singolo punto di attacco, il che rende i tentativi di *hackerare* tali sistemi troppo costosi e futili.

Sistema centralizzato e sistema distribuito





Tecnologia Blockchain I

La *blockchain* è un tipo specifico di DLT in cui i dati non vengono salvati in un singolo file, ma le informazioni sono suddivise in blocchi multipli, collegati tra loro, in una struttura a catena.

Un DLT generico consente tipicamente quattro operazioni: *Create*, *Retrieve*, *Update* e *Delete*. Invece, una blockchain consente solo due operazioni: *Create* e *Retrieve*. **Nessun dato può essere modificato o eliminato**.

La blockchain è quindi una lista in continua crescita di blocchi interconnessi di dati. Ogni blocco ha un riferimento al blocco precedente. L'intera blockchain è in possesso di tutti gli utenti della rete, che sono collegati tra loro in una rete *peer-to-peer*.

Questa struttura garantisce l'integrità e l'immodificabilità del database, ma come assicurare che la blockchain sia e rimanga identica presso ciascuno dei nodi nella rete senza alcuna autorità centrale al potere?

Tecnologia Blockchain II

Una tale struttura dati distribuita è **per costruzione essenzialmente inattaccabile e non modificabile in modo fraudolento**, quindi è impossibile attaccare contemporaneamente migliaia di utenti senza lasciare tracce, anche se ogni singolo utente non ha sistemi di protezione sofisticati.

Al contrario, quando i dati sono in una singola copia nelle mani di un'autorità centrale di fiducia, **tutta la sicurezza del sistema si basa sul livello di protezione del server centrale dell'autorità** e sulla lealtà di coloro che lavorano per l'autorità.

Tecnologia Blockchain III

Se tutti possiedono il database, non c'è privacy e tutti sanno tutto di tutti.

Viceversa, quando i dati sono detenuti solo dall'autorità centrale, la privacy è garantita dalla fiducia che gli utenti pongono (talvolta erroneamente) nell'autorità centrale.

La situazione non è così negativa come appare a prima vista perché attraverso l'uso della crittografia, la privacy delle transazioni può essere garantita pur registrandole su un database condiviso e quindi completamente pubblico.

Tecnologia Blockchain IV

La caratteristica della **scarsità digitale** si basa invece sulla struttura della blockchain, in particolare sulla sua immutabilità.

Supponiamo che una transazione dica *A dà a B 1 bitcoin*: si consegna semplicemente un certo valore digitale a B che conferma che la transazione è avvenuta (esistono sistemi crittografici ben noti per farlo in modo sicuro, come le **firme digitali**). Quindi, B potrebbe usare questo valore più volte e spendere ripetutamente i soldi ricevuti da A (il cosiddetto **double spending**).

Questo problema è risolto grazie all'uso di un **protocollo di consenso** ed è la vera innovazione della tecnologia blockchain.

Tipi di Blockchain

Le blockchain possono essere divise in tre macrocategorie:

- **Public permissionless**: chiunque è libero di unirsi e partecipare alle attività principali della rete blockchain.
- **Private permissioned**: l'accesso è consentito solo a partecipanti selezionati e verificati; l'operatore ha il diritto di sovrascrivere, modificare o eliminare le voci sulla blockchain.
- **Public permissioned**: ha proprietà sia delle blockchain private che pubbliche.

Analizziamo tutte e tre più in dettaglio.

Public permissionless

Una blockchain pubblica è una blockchain in cui chiunque può leggere, scrivere e verificare le attività in corso sul *network*. Questo contribuisce a raggiungere la natura autogovernata e decentralizzata della blockchain stessa.

Il vasto numero di partecipanti alla rete blockchain sicura mantiene al sicuro da violazioni dei dati, tentativi di *hacking* o altri problemi di sicurezza informatica. Più partecipanti ci sono, più sicura è una blockchain.

Uno degli svantaggi principali contestati alle blockchain pubbliche è il pesante consumo energetico necessario per mantenerle. Tuttavia, non tutte le blockchain consumano enormi quantità di elettricità.

Altri problemi includono la mancanza di completa privacy e anonimato.

Bitcoin e **Ethereum** sono le due blockchain pubbliche più importanti.

Private permissioned

I partecipanti possono unirsi a una rete blockchain privata solo tramite un invito in cui la loro identità è verificata. La convalida è effettuata dall'operatore della rete, o mediante un protocollo chiaramente definito.

Una blockchain privata non è decentralizzata. È un registro distribuito che funziona come un database chiuso, protetto con concetti crittografici e in base alle esigenze dell'organizzazione. Le blockchain private danno priorità all'efficienza a discapito dell'immodificabilità.

Pur essendo progettate appositamente per applicazioni aziendali, le blockchain private perdono molte delle preziose caratteristiche dei sistemi senza permessi semplicemente perché non sono applicabili. Inoltre, sono soggette a violazioni dei dati e ad altre minacce alla sicurezza.

Public permissioned

Le blockchain permissioned sono un mix tra le blockchain pubbliche e private, e supportano molte opzioni di personalizzazione.

Chiunque può unirsi alla rete, dopo un idoneo processo di verifica dell'identità. Alcune concedono autorizzazioni speciali e designate per eseguire solo attività specifiche in una rete.
Le blockchain permissioned consentono il concetto di *Blockchain-as-a-Service*, una blockchain progettata per essere scalabile per le esigenze di molte aziende o compiti che i *provider* affittano ad altre imprese.

Gli svantaggi delle blockchain permissioned rispecchiano quelli delle blockchain pubbliche o private, a seconda di come sono configurate.

Un buon esempio di blockchain pubblica permissioned è **Ripple**.

Protocolli di consenso

Protocollo di Consenso I

Un problema che deve necessariamente essere affrontato e risolto da ogni blockchain è quello di determinare **chi inserisce i nuovi blocchi, e come vengono inseriti**.

Dal momento che una blockchain è un registro distribuito e **non esiste una copia principale della catena**, è necessario che i nodi della rete si accordino sulle informazioni con cui aggiornare il registro.

Questo processo decisionale avviene attraverso l'esecuzione di un **protocollo di consenso**

Protocollo di Consenso II

Un protocollo di consenso è un protocollo che viene eseguito dai nodi di una rete per accordarsi su una **visione condivisa** in un contesto in cui i nodi della rete non si fidano gli uni degli altri (alcuni nodi potrebbero essere **malfunzionanti** oppure **malevoli**).

Protocollo di Consenso III

Esistono diverse tipologie di protocolli di consenso che permettono ai nodi di una blockchain di accordarsi sui nuovi blocchi da inserire a registro.

Il primo dei sistemi che è stato ideato per creare un protocollo di consenso è la **Proof of Work** (PoW). Che consiste nel permettere ai nodi della blockchain di poter proporre un blocco da inserire a registro solo dopo aver risolto un crittopuzzle.

Proof of Work (PoW) I

La forma più semplice di PoW è realizzata attraverso l'uso di una funzione hash, e può essere descritta nel seguente modo:

1. Dato un qualsiasi testo, colui che esegue la PoW aggiunge alcuni caratteri scelti a caso (**nonce**) e calcola l'hash del testo ottenuto.
2. Se il risultato dell'hash è inferiore a un certo valore (**target**), la PoW è completata e l'hash calcolato può essere utilizzato come prova del lavoro svolto.
3. Se, d'altra parte, il risultato dell'hash è maggiore del target, si ricomincia dal punto 1.

Esempio: il testo di partenza è "Filippo" e inizio la proof of work:

- calcolo l'hash di "Filippo**£T3**" $\rightarrow$ 0010101.... $>$ target;
- calcolo l'hash di "Filippo**Y98**" $\rightarrow$ 0001011.... $>$ target;
- ...
- calcolo l'hash di "Filippo**AFZ**" $\rightarrow$ 0000001.... $<$ target !

Proof of Work (PoW) II

Vista la natura delle funzioni hash, non è possibile scegliere ad hoc i caratteri da inserire nel testo per ottenere un valore di hash piccolo.

Quindi, **l'unico modo per procedere è inserire caratteri a caso** e riprovare finché non si ottiene un hash inferiore al target (che inizia con **un numero sufficientemente grande di zeri**).

Proof of Work (PoW) III

Il primo nodo della rete che **risolve il crittopuzzle** può proporre il proprio blocco, e **se il blocco è valido** verrà accettato da tutta la rete!

Ogni volta che viene inserito **un nuovo blocco**, viene proposto **un nuovo crittopuzzle** per inserire il blocco successivo.

Il nuovo cryptopuzzle viene usato per stabilire chi genererà il blocco successivo e così via...

Proof of Work (PoW) IV

Scegliendo opportunamente il target, si può rendere la PoW più o meno sfidante.

E' fondamentale trovare il giusto **compromesso** tra:

- una sfida troppo semplice $\rightarrow$ alta probabilità di avere tanti vincitori quasi simultanei della PoW.
- una sfida troppo complessa $\rightarrow$ una produzione di blocchi troppo lenta.

Proof of Work (PoW) V

Date le caratteristiche casuali del problema, è possibile stimare facilmente, fissata la potenza computazionale messa a disposizione, quanto tempo ci vorrà in media per produrre una PoW valida, e quindi per ottenere una nuova proposta di blocco.

Per esempio, il compromesso scelto da Bitcoin è 10 minuti come valore atteso di tempo per risolvere la PoW.

NB: **trovare una soluzione può essere un processo laborioso, ma verificare la soluzione è praticamente immediato!**

Proof of Work (PoW) VI

Nelle prissime lezioni si scenderà più nel dettaglio del protocollo di Bitcoin.

I protocolli PoW sono vulnerabili a un attacco noto come **attacco del 51%**. Questo attacco avviene quando un gruppo di utenti controlla più del 50% della potenza di calcolo della rete. Almeno teoricamente, questi utenti hanno il potere di alterare la blockchain.

Ad esempio, gli attaccanti sarebbero in grado di **impedire l'inserimento di alcune transazioni**, o di **cancellare alcuni blocchi in cima alla blockchain** ricreandone altri diversi. Annullare le transazioni potrebbe consentire loro di spendere due volte le monete (**double spending**), uno dei problemi che i meccanismi di consenso come la PoW sono stati creati per prevenire.

Proof of Stake (PoS) I

La **Proof of Stake** (PoS) è il secondo protocollo di consenso più importante.

Il principio è che i blocchi vengano inseriti tenendo conto della **quantità di criptovaluta posseduta dal validatore del blocco**, con l'idea che coloro che detengono grandi quantità di valuta abbiano interesse a far funzionare il sistema.

Proof of Stake (PoS) II

Le soluzioni adottate dalle diverse blockchain per tenere conto della quantità di moneta detenuta sono diverse:

- **Casuale**: Alcune criptovalute utilizzano sistemi casuali, in cui una maggiore quantità di moneta detenuta aumenta la probabilità di diventare un validatore.

- **Antichità**: le crittovalute possedute per molto tempo garantiscono una maggiore probabilità di essere selezionati come validatori. L'antichitò si azzera quando la moneta permette la nomina a validatore o se si supera una certa soglia.

Proof of Stake (PoS) III

I vantaggi e gli svantaggi di PoS rispetto a PoW sono i seguenti:

- Il consumo energetico è molto inferiore.
- È più veloce, e il tempo di inserimento dei blocchi è costante.
- I minatori di una blockchain PoW sono meno coinvolti rispetto ai validatori di una blockchain PoS.

Proof of Authority I

Nella **Proof of Authority** (PoA), ciò che si mette in gioco non è la valuta (come nella PoS), ma la reputazione. Il termine è stato proposto nel 2017 dal co-fondatore ed ex-CTO di Ethereum Gavin Wood.

Ovviamente, per fare questo è necessario che i validatori rendano pubblica la loro identità. Il modello *Proof of Authority* **si basa su un numero limitato di validatori**, fattore che lo rende un sistema altamente scalabile.
L'idea è di pre-selezionare dei nodi affidabili e quindi delegare a loro l'inserimento dei nuovi blocchi.

Proof of Authority II

L'algoritmo di consenso PoA necessita di:

- identità valide e affidabili dei validatori;
- difficoltà nel diventare e rimanere un validatore.

Un possibile difetto è che, essendo i validatori pochi e conosciuti, un attaccante potrebbe cercare di corromperli per compromettere il sistema.

Nella PoA c'è alta scalabilità, ma la decentralizzazione è fortemente limitata.

In genere, nei sistemi PoA pongono anche dei limiti al numero di blocchi che può inserire un singolo validatore, per esempio che non possa inserire blocchi consecutivi.

Protocolli Byzantine Fault Tolerant (BFT)

Il problema del consenso non ha origine con l'avvento delle blockchain; è in realtà un problema classico nell'informatica e nelle telecomunicazioni.

Uno dei lavori più importanti in questo ambito è un paper di Leslie Lamport, Robert Shostak e Marshall Pease, del 1982 in cui introducono il **Problema dei Generali Bizantini**.

I protocolli **Byzantine Fault Tolerant** (BFT) sono in grado di risolvere il problema dei Generali Bizantini e possono essere usati anche come protocolli di consenso per blockchain.

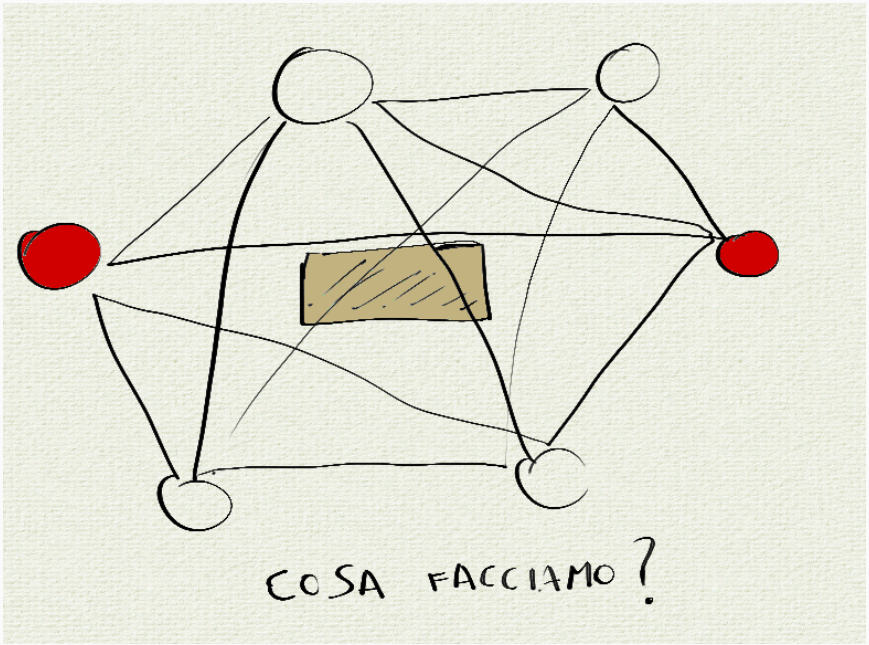
Il Problema dei Generali Bizantini I

Il Problema dei Generali Bizantini viene descritto come segue:

- L'esercito Bizantino è composto da alcune divisioni che accerchiano la città dei nemici;
- le divisioni dell'esercito possono comunicare tra loro solo tramite l'utilizzo di messaggeri;
- dopo aver osservato il nemico devono accordarsi su un piano d'azione **comune**.

Il Problema dei Generali Bizantini II

Tuttavia, alcuni dei generali che sono a capo delle divisioni dell'esercito **sono traditori**, e proveranno a ostacolare i generali onesti nel raggiungimento del consenso sul piano d'azione.



Il Problema dei Generali Bizantini III

Ogni generale inizia il protocollo con **una propria opinione sul piano d'azione**.

I generali devono quindi utilizzare un protocollo che garantisca le seguenti richieste:

- Alla fine dell'esecuzione del protocollo, **i generali onesti saranno d'accordo sul piano d'azione**;
- Un piccolo numero di traditori non può convincere gli altri generali ad **adottare un piano "sbagliato"**.

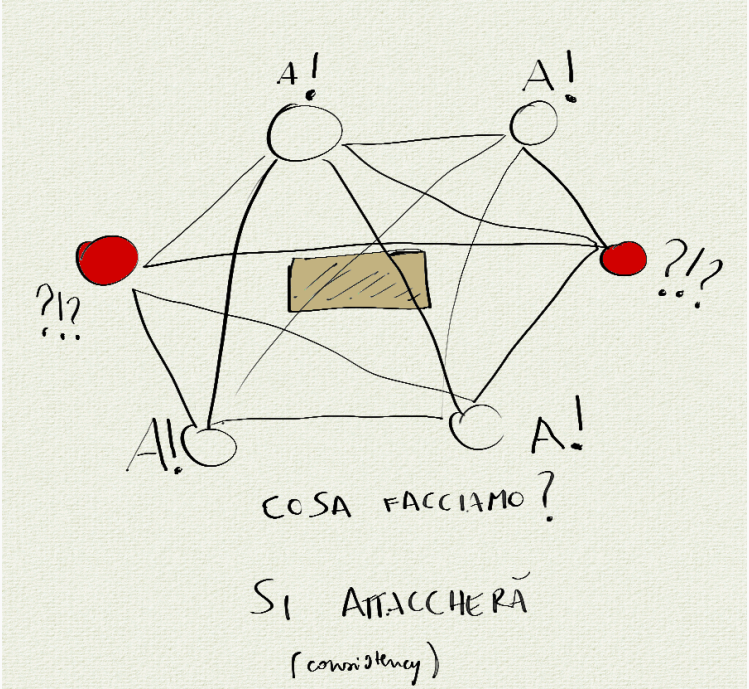
Il Problema dei Generali Bizantini IV

In letteratura queste proprietà sono definite come

- **agreement**: il piano di azione di tutti i generali onesti *è lo stesso*.
- **consistency**: se all'inizio del protocollo *tutti i generali onesti hanno la stessa opinione* sul piano d'azione, i generali onesti *raggiungeranno il consenso su quel piano.*

Il Problema dei Generali Bizantini V

La seconda condizione sta a significare che i generali malevoli non sono in grado di far convergere il consenso su una soluzione "sbagliata".



I protocolli che soddisfano le condizioni di **agreement** e **consistency** sono detti protocolli **Byzantine Fault Tolerant (BFT)** o **Byzantine Agreement (BA)** protocols.

BFT: un (cattivo) esempio giocattolo I

Per evidenziare maggiormente le difficoltà, analizziamo un esempio più specifico, con 11 generali.

Consideriamo il seguente protocollo giocattolo:

Ogni generale invia agli altri generali la propria opinione, che può essere o attacco o ritirata. L'output di ogni generale sarà il valore che è stato scelto dalla maggioranza dei generali.

BFT: un (cattivo) esempio giocattolo II

Questo protocollo è BFT assumendo che ci possa essere nella rete 1 solo nodo malevolo?

Osserviamo:

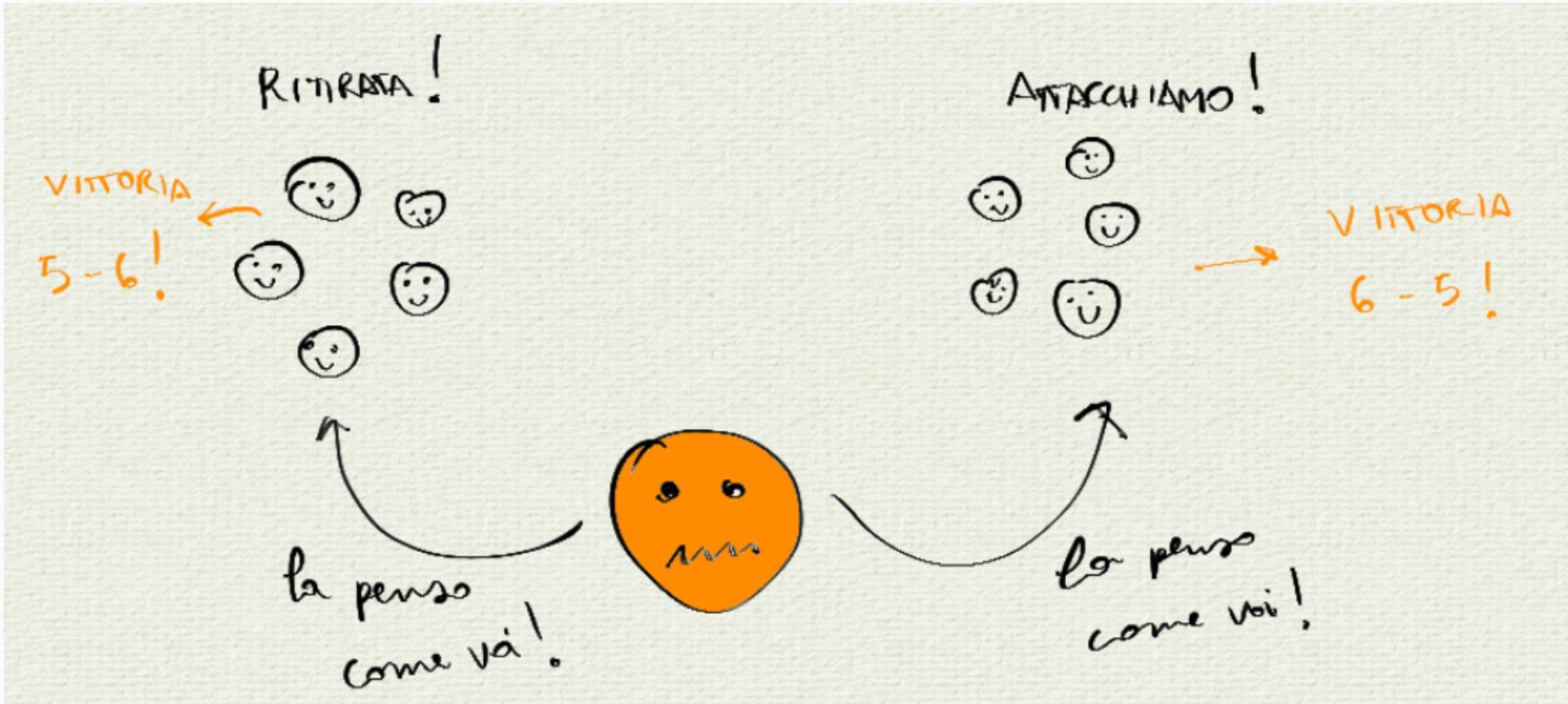
se fin dall'inizio c'è uno sbilanciamento di opinioni tra gli onesti (per esempio 6 inviano ritirata e 4 attacco), allora il nodo malevolo non può evitare il raggiungimento del consenso.

Supponiamo che 5 generali onesti abbiano proposto di voler attaccare e 5 invece di ritirarsi. **Il generale malevolo può rompere il consenso**!

- inviando a qualcuno un messaggio per la ritirata;
- inviando agli altri un messaggio per attaccare.

BFT: un (cattivo) esempio giocattolo III

Possiamo notare che in questo protocollo la **consistency è garantita**. Invece la condizione di **agreement** può essere infranta da un *singolo generale malevolo*.



BFT: un secondo (cattivo) esempio giocattolo

Consideriamo il seguente protocollo giocattolo: i valori che considereremo sono **attacco** o **ritirata** ed un valore di default **"?"** che si riferisce a una condizione di stallo. Il protocollo di comunicazione è il seguente:

Ogni generale invia la propria opinione, tra le tre disponibili, agli altri generali. L'output di ogni generale sarà "?".

In questo caso la condizione di **agreement** è soddisfatta in quanto l'output di tutti i generali onesti sarà lo stesso ("?"), però la proprietà di consistency non sarà soddisfatta perchè anche se tutti i generali sono d'accordo su attaccare o ritirarsi, comunque loro si accorderanno sul rimanere in una situazione di stallo.

In questo caso non è neanche necessaria la presenza di generali malevoli per rompere la *consistency*!

BFT: un terzo esempio

Nel primo approfondimento presenteremo un esempio di protocollo di consenso bizantino che è alla base del protocollo di consenso della blockchain di **Algorand**.

Differenze tra PoW e BFT

- Se il vincitore del *cryptopuzzle* è onesto, allora diffonderà il blocco a tutti i nodi della rete, e verrà raggiunto il consenso su quel blocco.

- Se il vincitore è malevolo potrebbe comunicare la soluzione solo ad alcuni, e gli altri riceverebbero il blocco del "secondo arrivato".

In questo caso non si avrebbe un consenso nella rete immediato, cosa che si ha con i protocolli BFT.

Fortunatamente questo stato di divisione all'interno della rete può essere risolto, come presenteremo nelle prossime lezioni.

Approfondimento: il protocollo di consenso di Algorand

Protocollo di consenso di Algorand

Algorand è una blockchain con protocollo di consenso **PoS che si basa sull'utilizzo di un protocollo di consenso BFT**.

Noi descriveremo una **versione semplificata del protocollo di consenso**, una versione in cui tutti i nodi prendono parte al raggiungimento del consenso e in cui supponiamo che **ogni utente controlli la stessa quantità di crittovaluta**.

Contesto semplificato

Assunzioni di rete: consideriamo

- un **network completo** in cui esiste un canale di comunicazione diretto tra ogni partecipante alla rete (non c'è bisogno di firmare messaggi per provare la paternità);
- esiste un **orologio globale** in cui ad ogni clock dell'orologio i partecipanti al protocollo inviano i loro messaggi;
- un network in cui i messaggi inviati raggiungono **istantaneamente** il destinatario.

Queste sono le caratteristiche di un **network sincrono e completo**.

Nodi Malevoli

Il protocollo che presenteremo è un protocollo BFT eseguito da n nodi, ed è in grado di resistere alla minaccia di t nodi con $t < \frac{n}{3}$: meno di un terzo della rete.

Recap: Il Problema dei Generali Bizantini II

Considereremo protocolli in cui ogni generale G_i inizia il protocollo con un **input** v_i.

Il protocollo terminerà con un **output** o_i per ogni generale G_i per cui:

- **agreement**: Esiste un valore o per cui per ogni generale G_i **onesto**, $o_i = o$;
- **consistency**: Se esiste un valore v per cui **ogni generale** G_i inizia con $v_i = v$, allora $o_i = v$ per ogni G_i onesto.

Obiettivo

L'obbiettivo dei nodi (o generali) è quello di raggiungere il consenso sul nuovo blocco da inserire a registro.

L'approccio di Algorand:

- i generali eleggono randomicamente un piccolo sottogruppo di **potenziali leader** che proporranno un loro blocco;
- raggiungono il consenso su **uno dei blocchi proposti** oppure il **blocco NULLO**.

Panoramica della soluzione

I protocolli che vengono eseguiti in ogni round per la creazione del nuovo blocco sono i seguenti:

- **selezione del leader** che propone il nuovo blocco attraverso una lotteria;
- **graded consensus**: un "sondaggio" preliminare utile ad ogni generale per capire se il consenso può essere raggiunto;
- **binary byzantine agreement**: un protocollo che finalizza il raggiungimento del consenso.

Selezione dei possibili leader I

L'idea che sta alla base della lotteria usata per selezionare i leader è la seguente:

- ogni nodo è associato ad una chiave pubblica di un algoritmo di **firma unica** SIG, ossia tale che dato un messaggio M e la chiave privata sk, esiste una sola firma $\sigma = SIG(M, sk)$ ad essi associata.

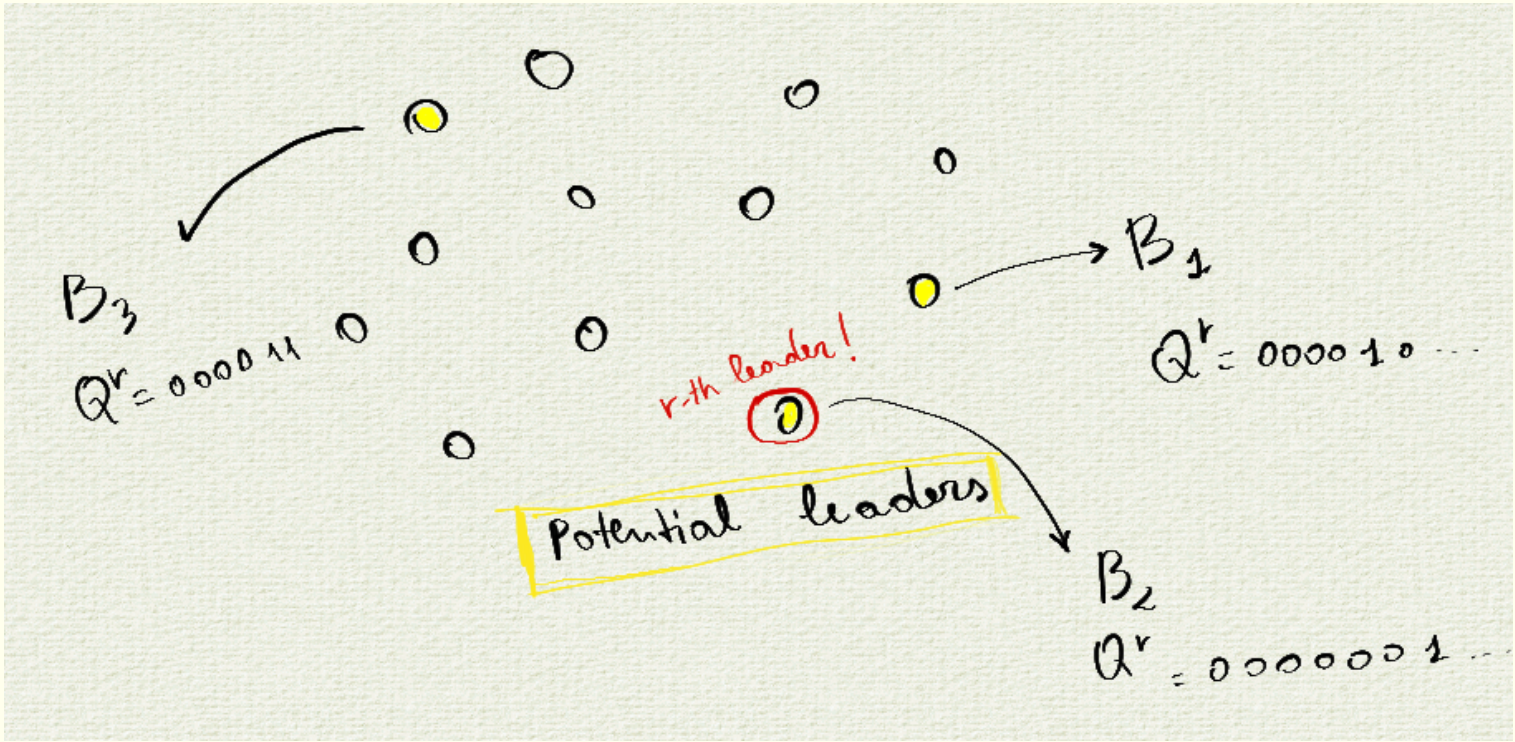
- ad ogni round r viene generato un nuovo **seed** Q^r che è incluso nel blocco creato.

  Con questi ingredienti è possibile costruire una lotteria per il round $r + 1$.

Selezione dei possibili leader II

All'inizio di ogni round $r+1$, ogni generale G_i:

- firma con la propria chiave privata il seed Q^r;
- se la firma è minore di una certa soglia allora G_i è un **potenziale leader** per il round $r+1$ e propone un blocco alla rete (pubblicando la sua firma).
- altrimenti osserva.



Selezione dei possibili leader III

La soglia sotto la quale un nodo è eletto a potenziale leader è scelta in modo tale che il numero di potenziali leader sia sufficiente grande da rendere estremamente improbabile che tutti siano malevoli (per garantire la **liveness della blockchain**).

NB: usiamo la firma unica per selezionare i potenziali leader per mantenere l'anonimato dei leader del round finchè loro non dichiarano di esserlo.

Ogni generale sceglie il proprio input

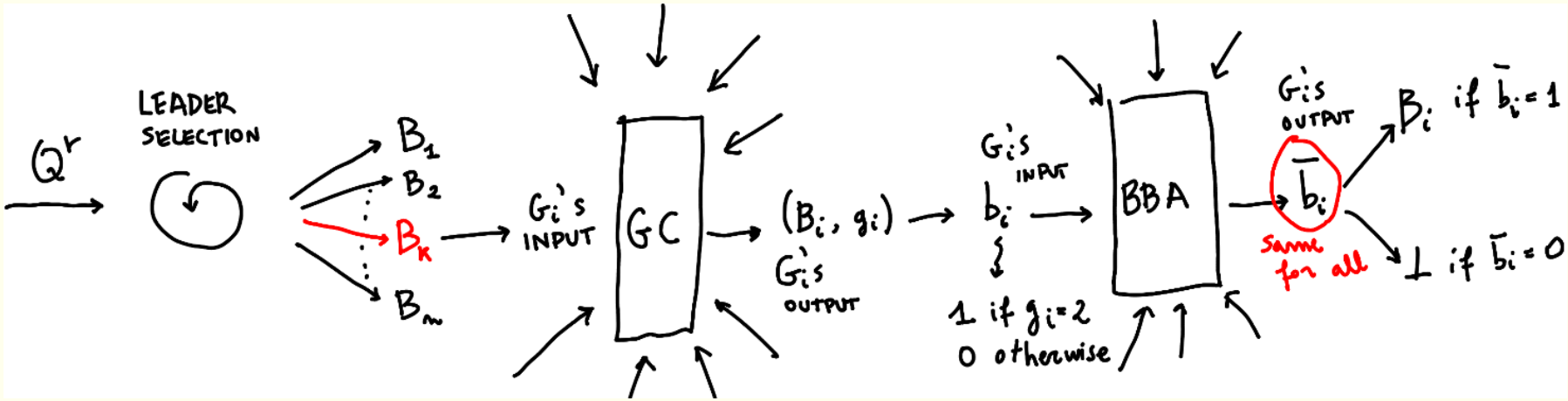
Pubblicati i blocchi dei potenziali leader, **ogni generale deve scegliere il proprio input del protocollo BFT**

ogni generale sceglierà il blocco del potenziale leader G_i per cui la firma di Q^r assume il valore più piccolo.

NB: se il nodo con la firma più piccola è un generale onesto, allora **tutti i generali onesti inizieranno il protocollo BFT con lo stesso input**. Altrimenti non è detto.

Flusso del protocollo

Ci troviamo alla selezione dell'input del protocollo GC da parte del generale G_i.



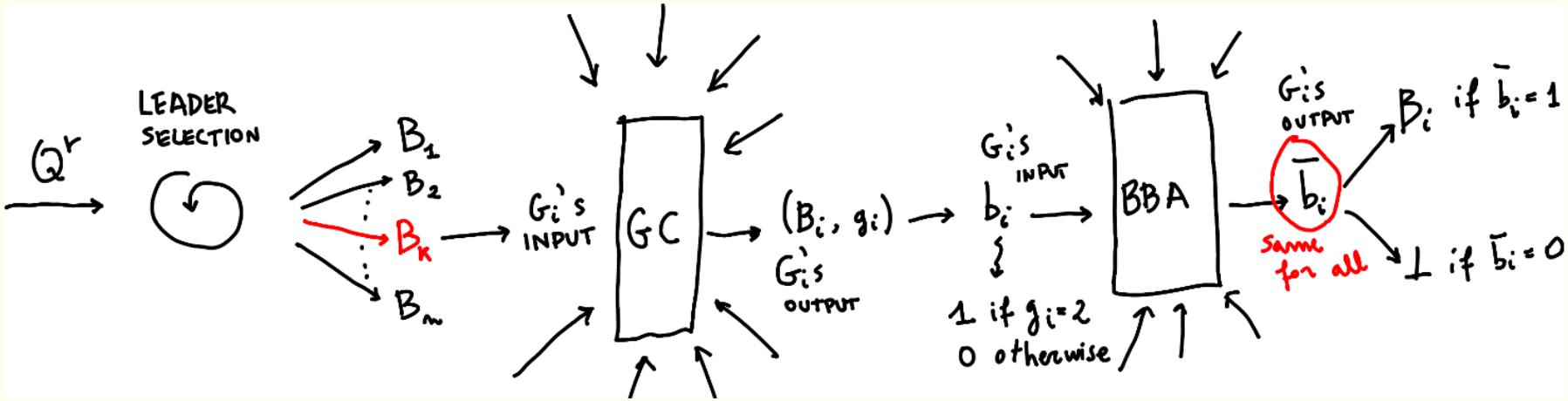
Graded Consensus (GC)

I generali iniziano un protocollo di due step che li porta ad ottenere ciascuno una visione **soggettiva e approssimativa** sullo stato del consenso in rete. In particolare

- l'input di ciascun generale sarà un blocco v_i';
- l'output del protocollo sarà della forma:
  - $(v_i, 2)$ se per G_i il blocco v_i è **molto diffuso** tra i generali;
  - $(v_i, 1)$ se per G_i il blocco v_i è **parzialmente diffuso** tra i generali;
  - $(\perp, 0)$ se per G_i esiste un **vasto disaccordo** nella rete sul blocco da aggiungere.

Flusso del protocollo

Ci troviamo nella generazione dell'output di GC e creazione dell'input di BBA.



Binary Byzantine Agreement (BBA)

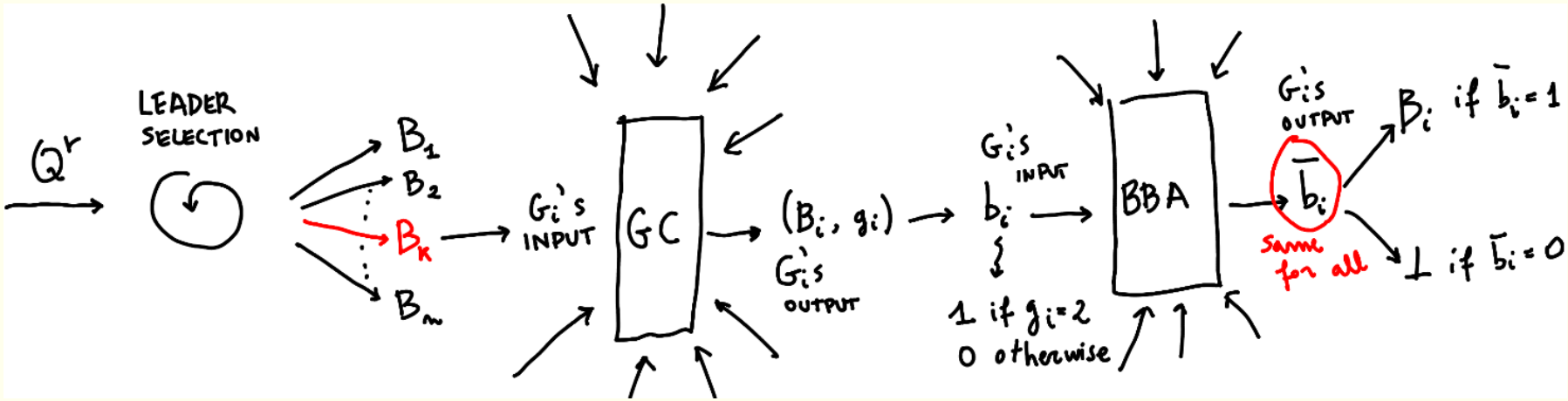
Se il generale G_i finisce il graded consensus con in output il grado 2, quindi è convinto che il blocco v_i sia il blocco scelto da gran parte della rete, inizierà **un protocollo di consenso binario** con in input il bit 1. Altrimenti inzierà col bit 0.

Per alcune proprietà del protocollo GC, vale che:

Se il consenso viene raggiunto sul bit 1 allora tutti i giocatori onesti avranno finito il protocollo GC **con lo stesso valore** v (vedremo perchè), e v sarà il nuovo blocco inserito a registro.

Flusso del protocollo

Ci troviamo fuori dal BBA



Scendiamo più nel dettaglio

$$\#_i^n(M)$$

rappresenta il numero di messaggi che il generale G_i ha ricevuto durante lo step n del protocollo conteneti il messaggio M.

Vediamo nel dettaglio come sono definiti i protocolli di **Graded Consensus** e di **Binary Byzantine agreement**.

Graded Consensus Protocol

Ogni generale G_i inizia il protocollo Graded Consensus con un valore $v_i' \in V$ in input.

- **STEP 1.** Ogni G_i invia v_i' ad ogni altro generale.
- **STEP 2.** Ogni generale G_i invia a tutti i generali la stringa $x \neq \perp$ se e solo se $\#_i^1(x) \geq \lfloor \frac{2n}{3} \rfloor + 1$. Altrimenti G_i invierà $\perp$.
- **OUTPUT DETERMINATION.** Ogni generale G_i invierà la coppia (v_i, g_i) calcolata come segue:
  - Se per qualche $x \neq \perp$, $\#_i^2(x) \geq \lfloor \frac{2n}{3} \rfloor + 1$, allora $(v_i, g_i) = (x, 2)$.
  - Se, per qualche $x \neq \perp$, $\#_i^2(x) \geq \lfloor \frac{n}{3} \rfloor + 1$, allora $(v_i, g_i) = (x, 1)$.
  - Altrimenti, $(v_i, g_i) = (\perp, 0)$.

Proprietà del graded consensus I

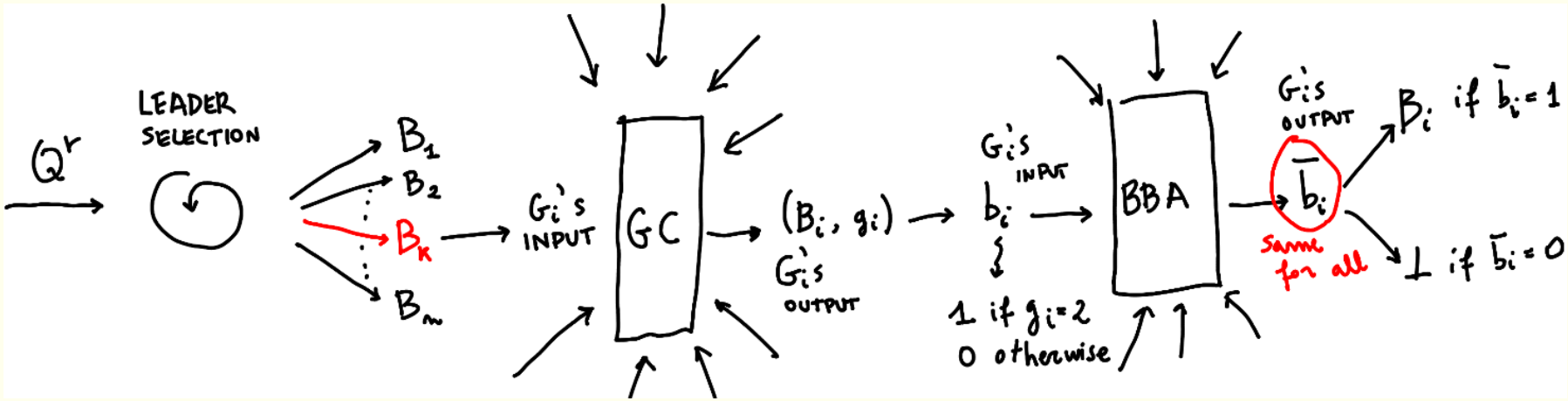
Il protocollo di graded consensus sopra descritto soddisfa le seguenti proprietà:

- **per ogni coppia di generali onesti G_i, G_j vale $|g_i - g_j| \leq 1$:** non può succedre che un onesto setti il grado a 0 e un altro a 2.
  N.B.: i generali malevoli contano meno di $\frac{1}{3}$ della rete.

- **per ogni coppia G_i, G_j di generali onesti, se $g_i, g_j \geq 1$ allora $v_i = v_j$:**
  (discusso di appendice)

- **se tutti i generali onesti iniziano il protocollo GC con lo stesso input v', allora l'output di ogni onesto sarà $(v', 2)$.**
  N.B.: tutti gli onesti nel secondo step del protocollo invieranno in messaggio v'.

Flusso del protocollo



Binary Byzantine Agreement (BBA) I

Ogni generale G_i inizia il protocollo Binary Byzantine Agreement con un bit b_i in input sulla base dell'output di GC.

- **STEP 1**. [**Coin-Fixed-To-0 Step**] Ogni generale G_i invia b_i.
  1. se $\#_i^1(0) \geq 2t+1$, allora Gi setta $b_i = 0$, invia $(0, \star)$, ritorna come output $out_i = 0$, e HALTS.
  2. se $\#_i^1(1) \geq 2t+1$, allora G_i setta $b_i = 1$.
  3. Altrimenti G_i setta $b_i = 0$.
- **STEP 2**. [**Coin-Fixed-To-1 Step**] Ogni generale G_i invia b_i.
  1. se $\#_i^2(1) \geq 2t+1$, allora G_i setta $b_i = 1$, invia $(1, \star)$, ritorna come output $out_i = 1$, e HALTS.
  2. se $\#_i^2(0) \geq 2t+1$, allora G_i setta $b_i = 0$.
  3. altrimenti i setta $b_i = 1$.

Se G_i non si è fermato in nessuno di questi due step allora...

Binary Byzantine Agreement (BBA) II

STEP 3. [**Coin-Genuinely-Flipped Step**] Ogni generale G_i invia b_i e $SIG_i(r, \gamma)$.

1. se $\#_i^3(0) \geq 2t + 1$, allora i setta $b_i = 0$.
2. se $\#_i^3(1) \geq 2t + 1$, allora i sets $b_i = 1$.
3. Altrimenti, essendo S_i l'insieme dei generali che hanno inviato a G_i un messaggio valido durante STEP 3, G_i calcola $k = H(min_{j \in S_i} H(SIG_j(Q^r, \gamma))$ e setta $b_i = k_1$, il primo bit del digest k; aumenta γ_i di 1 e torna allo STEP 1.

Se per esempio $k = 00000011...101$ è la firma più piccola ricevuta da un generale, se entra nella clausola 3 allora setterà il bit a 1 (l'ultimo bit della firma).

Cosa vogliamo provare

Il protocollo BBA è un protocollo BFT binario, quindi in cui il consenso può essere raggiunto solo sui valori 0 e 1.

Le proprietà che devono essere soddisfatte sono

- **consistency**: *banale da verificare in quanto se tutti i generali onesti iniziano il protocollo con valore 0 o con valore 1 allora tutti i generali onesti termineranno il protocollo alla prima iterazione in step 1 o step 2 rispettivamente.*
- **agreement**: *Non banale. Discusso in appendice.*

Per provare che vale la proprietà di agreement evidenziamo alcune caratteristiche di questo protocollo (discusse in appendice).

Proprietà di BBA I

Diciamo che i generali **raggiungono l'agreement** quando il bit che hanno salvato e aggiornano durante l'esecuzione del protocollo è lo stesso per tutti. Anche se non hanno raggiunto l'halt!

Si può mostrare che valgono le seguenti proprietà:

1. l'**agreement** una volta raggiunto si mantiene;

2. quando **un onesto raggiunge l'halt**, allora è stato raggiunto l'agreement su quel bit.

3. ad ogni round i generali onesti raggiungono l'agreement con probabilità $> \frac{1}{3}$;

Proprietà di BBA II

Quindi la proprietà di **agreement** è implicata dal fatto che:

- quando un onesto esce dal protocollo, gli onesti sono in agreement (proprietà 2);
- il protocollo termina sempre visto che
  - l'agreement è raggiunto con probabilità $> \frac{1}{3}$ ad ogni round (Proprietà 3);
  - e l'agreement si mantiene una volta raggiunto (Proprietà 1).

Valendo l'agreement e la consistency, il **protocollo BBA è effettivamente un protocollo BFT**.

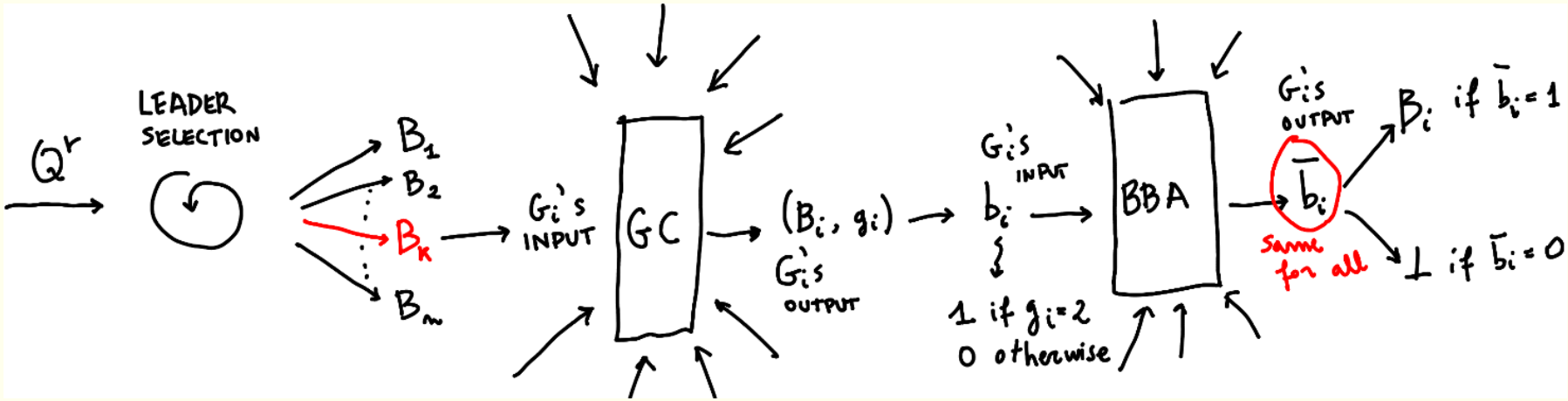
Mettiamo insieme i pezzi

Noi useremo i protocolli per la **selezione del leader, GC e BBA** per ottenere un protocollo BFT per valori qualsiasi!

Ricapitoliamo la struttura del protocollo:

- ad ogni round **vengono selezionati i potenziali leader** che creano una proposta di blocco e la diffondono;
- ogni generale **seleziona il blocco da supportare** sulla base dei blocchi ricevuti e inizia il GC;
- il GC restituisce una **panoramica dello stato di consenso** della rete e in particolare fornisce ai generali l'input con cui inziare il protocollo di Binary Byzantine agreement con cui i generali finalizzano il consenso;
- Se **il consenso è raggiunto su 1 allora viene finalizzato il blocco** v, parte dell'output del GC per ogni generale onesto, se **l'output è 0 allora viene finalizzato il blocco NULLO**.

Flusso del protocollo



Osservazione cruciale

E' interessante notare che se almeno un generale onesto termina il GC con output $(v, 2)$ come grado, allora tutti i generali onesti fanno rifermiento allo stesso blocco v.

L'input 1 nel BBA può essere quindi interpretato come un segnale che **tutti i generali onesti stanno facendo riferimento allo stesso blocco e quindi possono raggiungere il consenso su di esso**.

Il protocollo nel complesso è BFT

Questo è effettivamente un protocollo di consenso BFT per valori arbitrari. In particolare sono soddisfatte le proprietà:

- **consistency**: se tutti i generali onesti iniziano il protocollo con lo stesso valore v termineranno GC con $(v, 2)$ quindi inizieranno il BBA con 1 e terminerà con 1 portandoli a finalizzare v.
- **agreement**: se il consenso nel BBA viene raggiunto su 0 allora tutti gli onesti aggiungeranno il blocco di default $\perp$ al registro. Se viene raggiunto su 1, vuol dire che
  - almeno un onesto ha iniziato il BBA con 1 in input;
  - quindi quell'onesto ha finito il GC con grado 2;
  - quindi tutti gli onesti hanno finito il GC con riferimento allo stesso blocco v;
  - quindi tutti gli onesti aggiungeranno il blocco v al registro.

  quindi per le proprietà del GC tutti gli onesti hanno finito il GC

Proof of Stake

Questo protocollo può essere trasformato in una proof of stake:

- chi possiede k unità di crittovaluta ha la possibilità di creare k biglietti della lotteria nella selezione dei potenziali leader;
- chi possiede k unità di crittovaluta ha la possibilità di inviare k messaggi in tutti gli step successivi.

.

In questo caso l'assunzione che si fa è che meno di $\frac{1}{3}$ delle unità di crittovaluta siano controllate da utenti malevoli.

Appendice (Algorand)

Appendice A: Proprietà del graded consensus I

Per ogni coppia G_i, G_j di generali onesti, se $v_i, v_j \geq 1$ allora $v_i = v_j$.

Infatti, supponendo che esistano due generali onesti G_i, G_j per cui $g_i, g_j \geq 1$ e $v_i \neq v_j$ allora significa che, dopo il secondo step,

- G_i ha ricevuto almeno $\lfloor \frac{n}{3} \rfloor + 1$ messaggi per v_i
- G_j $\lfloor \frac{n}{3} \rfloor + 1$ messaggi per v_j.

Questo significa che, essendo $t < \frac{n}{3} \leq \lfloor \frac{n}{3} \rfloor + 1$ almeno un onesto nel secondo step ha inviato il messaggio v_i e un onesto ha inviato il messaggio v_j.

Questo significa che nel primo step questi due generali onesti hanno ricevuto più di $\lfloor \frac{2n}{3} \rfloor + 1$ messaggi uno per v_i e l'altro per v_j. Ma $2(\lfloor \frac{2n}{3} \rfloor + 1) > \frac{4n}{3} > n + t$. **Questo significa che un onesto ha inviato messaggi diversi a generali diversi. Assurdo.**

Appendice B: Proprietà di BBA I

Una volta raggiunto l'agreement, questo si mantiene negli step successivi.

Può essere facilmente osservato dalla descrizione degli step e dal fatto che i generali onesti sono più di $\frac{2n}{3}$.

Inoltre, una volta che i generali onesti settano lo stesso bit b alla fine di un qualsiasi step, non appena si arriva al successivo coin-fixed-to-b step, i generali raggiungono l'HALT e chiudono l'esecuzione del protocollo avendo raggiunto il consenso su b.

Appendice B: Proprietà di BBA II

Se in uno step un generale onesto raggiunge l'HALT, allora alla fine dello step tutti i generali onesti saranno in agreement:

se un generale onesto raggiunge l'HALT ha ricevuto $> \lfloor \frac{2}{3} \rfloor + 1$ messaggi per b quindi è impossibile che un altro onesto abbia ricevuto $> \lfloor \frac{2}{3} \rfloor + 1$ per $b - 1$. Ma essendo in un Coin-Fixed-To-b step, allora gli onesti che non raggiungono l'HALT settano il bit a b.

Appendice B: Proprietà di BBA III

Se all'inizio del terzo step nessun generale si è fermato, con probabilità almeno $\frac{1}{3}$ i generali onesti saranno in agreement (e si fermeranno nel round successivo). Nel terzo step ogni generale

eseguirà una delle 3 clausole prescritte dal protocollo di STEP 3.

Notate che i generali onesti **aggiornano il proprio bit** nelle seguenti possibili configurazioni:

- tutti entrano nella clausola c, $c \in \{1, 2, 3\}$;
- alcuni nella clausola $c \in \{1, 2\}$ e tutti gli altri nella clausola 3.

Proprietà di BBA IV

- **tutti gli onesti aggiornano il bit nella clausola c, $c \in \{1, 2, 3\}$:**
  Se i generali aggiornano il bit nella clausola 1 (2) allora raggiungono l'agreement sul bit 0 (1), se tutti lo aggiornano nella clausola 3 allora, se il random coin è associato ad un utente onesto (il che accade con probabilità $> \frac{2}{3}$), i generali onesti raggiungeranno l' agreement.

- *alcuni generali onesti aggiornano il bit nella clausola $c \in \{1, 2\}$ e tutti gli altri nella clausola 3*:
  con probabilità $> \frac{2}{3}$ il random coin è associato ad un onesto e con probabilità $> \frac{1}{2}$ farà settare ai generali onesti al bit $c - 1$ (0 se gli altri onesti lo hanno settato a 0 e 1 se gli altri lo hanno settato a 1).

Approfondimento di crittografia

Assunzioni

Uno schema crittografico è sempre basato su un'assunzione ritenuta *difficile*, nel senso che deve essere immediato calcolare la chiave pubblica a partire da quella privata, mentre l'algoritmo inverso è un problema che richiede anni e anni di calcolo.

Due esempi famosi sono:

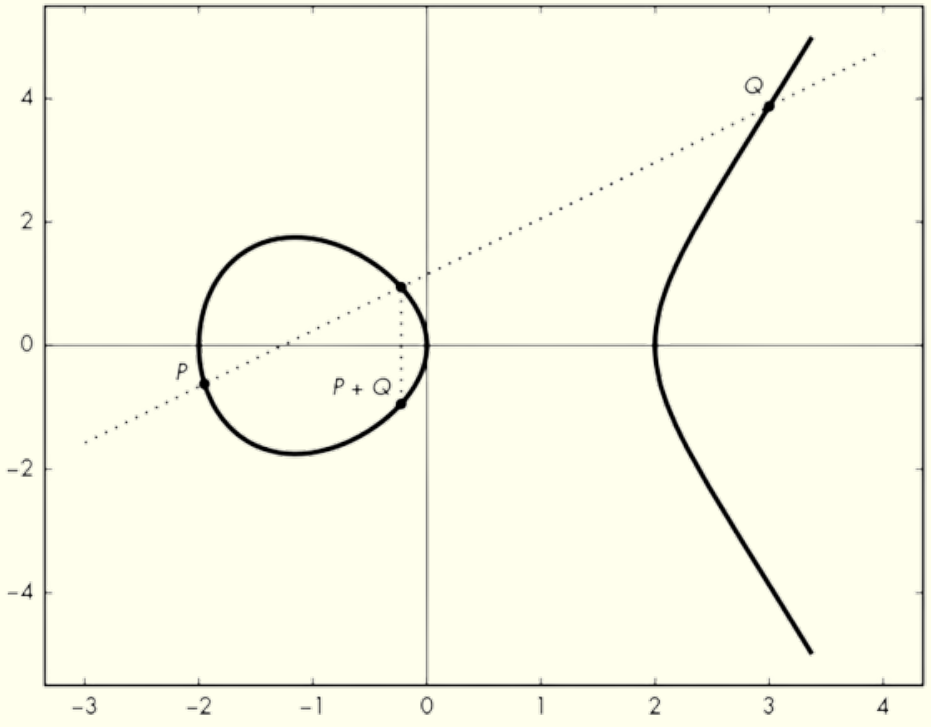
- **Problema della fattorizzazione**: dato $N = pq$, con p e q primi, trovare p e q.
- **Problema del logaritmo discreto**: dati due elementi dello stesso gruppo g e $y = g^x$, trovare x.

In questi casi, i primi p e q o l'esponente x rappresentano la *chiave privata*, mentre i numeri N e y sono le chiavi pubbliche.

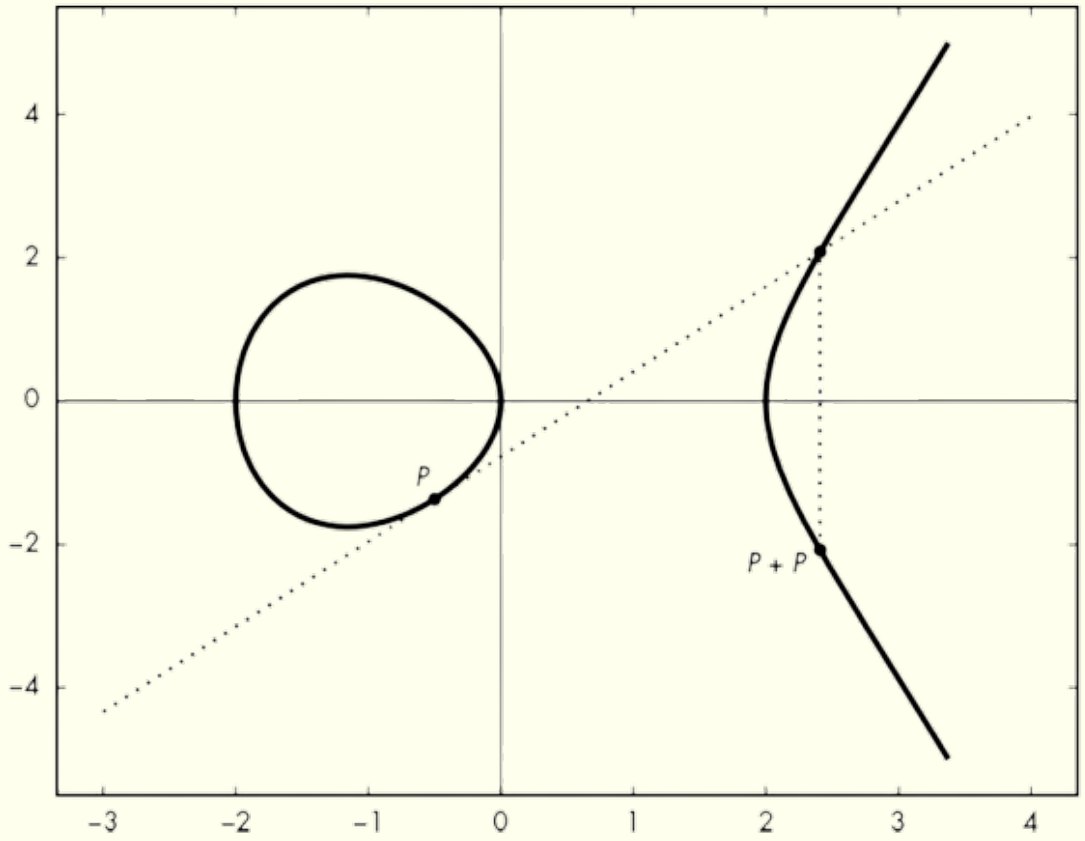
ECC I

Abbiamo visto che gli schemi di crittografia che ci interessano sono basati su curve ellittiche. Approfondiamo un po' l'argomento.

Una curva ellittica è composta dai valori di $\mathbb{Z}_q$ (l'insieme dei numeri tra 0 e $q-1$) che verificano un'equazione del tipo $\mathbf{y^2 = x^3 + ax + b}$ (possono essere utilizzate anche altre forme).
I punti P e Q di una curva possono essere sommati:



È anche possibile calcolare $2P = P + P$, e quindi kP, nel seguente modo:



ECC III

I grafici che sono stati mostrati per comprendere le operazioni sui punti di una curva sono stati costruiti disegnando tutti i punti nel piano reale che verificano l'equazione $y^2 = x^3 + ax + b$.

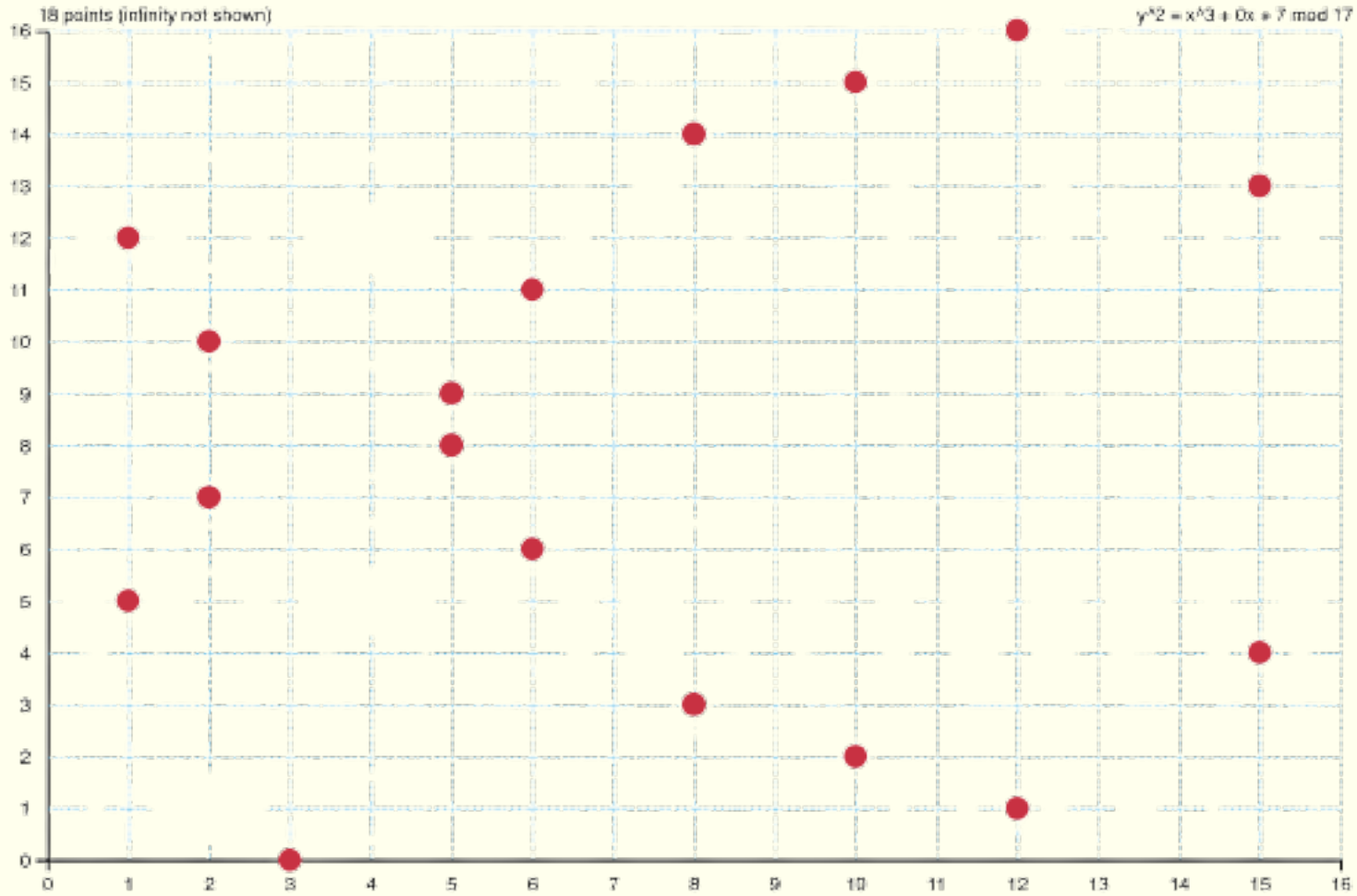
Tuttavia, nelle applicazioni crittografiche i valori di x e y che verificano l'equazione vengono presi nel campo $\mathbb{Z}_q$, quindi l'equazione è verificata modulo q.

Ad esempio, se consideriamo la curva ellittica $y^2 = x^3 + 7$, con $q = 17$, otteniamo uno spazio di 18 punti (17 rappresentati nell'immagine, più un punto speciale noto come il *punto all'infinito*).

Per disegnare curve ellittiche: Draw Elliptic Curves over Finite Fields.

ECC IV

Draw the elliptic curve $y^2 = x^3 + ax + b \mod r$, where a: 0 b: 7 r: 17



ECC V

Non è una buona idea sommare il punto P k volte per calcolare kP: *l'Algoritmo di Esponenziazione Veloce* (detto anche **Double-and-Add**), è molto più efficiente.
Ad esempio, possiamo calcolare $9P$ calcolando $2P$, $4P = 2P + 2P$ e quindi $8P = 4P + 4P$ e infine $9P = 8P + P$, effettuando 4 somme invece di 8.

L'analogo del problema del calcolo del logaritmo discreto in $\mathbb{Z}_q$ sulle curve ellittiche diventa determinare k, dati i punti P e $Q = kP$. Questa assunzione è il cosiddetto **Problema del Logaritmo Discreto su Curve Ellittiche** (ECDLP).

L'ECDLP è un problema considerato più difficile del DLP classico: è quindi possibile utilizzare chiavi più corte, mantenendo il livello di sicurezza che il DLP classico fornisce con numeri molto più grandi.

ECC VI

Data una curva ellittica e un insieme $\mathbb{Z}_q$ su cui è definita (con q molto grande, per rendere il sistema inattaccabile da attacchi di tipo **brute force**), e fissato un punto G della curva, la creazione di una coppia di chiavi pubblica/privata è molto semplice:

- La chiave segreta è un numero casuale d, compreso tra 0 e $q - 1$.
- La chiave pubblica viene quindi calcolata come $P = dG$.

È quindi semplice calcolare la chiave pubblica dalla chiave privata, mentre tornare indietro dalla chiave pubblica alla chiave privata è invece davvero difficile, poiché si presume che il problema del logaritmo discreto su curve ellittiche sia difficile da risolvere se il numero di punti della curva è abbastanza grande.