introduzione a Bitcoin

e alle criptovalute digitali

P.Bertoni

4 luglio 2014

Tabella Contenuti

contenuto

Criptovaluta
specifiche sul protocollo

Problema

implementare una valuta economica

- sicura
- basata su informatica
- decentralizzata
- distribuita

Criptovaluta

specifiche sulle transazioni

Problema

trasmettere transazioni di valuta tra enti

- pubbliche
- anonime $\Rightarrow$ tra **indirizzi**, non tra utenti
- autenticate
- non ripudiabili
- irreversibili

registrate in una sorta di storico globale

Transazione $\mathfrak{T}$

idea astratta

- unica modalità di circolazione della valuta
- atto tra *N* mittenti e *M* destinatari
- gli enti sono *indirizzi*, **non** persone!
- utenti incoraggiati a usare un indirizzo unico $\forall$ loro $\mathfrak{T}$
- $\mathfrak{T}$ paragonabile a un assegno
  "si certifica che, in data t,
  $\{x\}$ *ha versato* tot ₿ *a* $\{y\}$*, che ora ne è proprietario."*

gestione del resto

- $y' \in \{y\}$ destinatari, indirizzo controllato da chi emette $\mathfrak{T}$
- diversi resti frammentati sono riuniti come molteplici mittenti

Criptovaluta
specifiche sull'affidabilità

Problema

progettare un algoritmo di mining per convalidare transazioni

- *trattabile* da decidere
- *intrattabile* da risolvere
- dipendente da lista transazioni in attesa

motivazione a partecipare al mining

- ricompense
  - immediate
  - durevoli
- complessità lavoro onesto $\equiv$ complessità disonesto
- ma Pr[successo lavoro disonesto] $\rightarrow 0$

contenuto

Differenze con valute tradizionali

ontologia

esplicita: come unità fisiche — implicita: in funzione di transazione

---

fiducia nell'accettazione di moneta

difficoltà di contraffazione

---

possibilità di furto o smarrimento

gettone fisico — chiave privata di firma digitale

---

fiducia nel protocollo di supporto

ente nazionale o sovranazionale — modello formale di sicurezza

i primordi: eCash [Chaum]

sistema di firma digitale a *conoscenza zero*

Scenario (*e.g.* voto digitale)

- sia m plaintext, A il suo autore e F il firmatario
- $A \neq F$
- firma *classica*: $S(m, K^A_{PR})$
- firma *cieca*: $S^B(f(m), K^F_{PR})$
- F non può calcolare m data $f(m)$

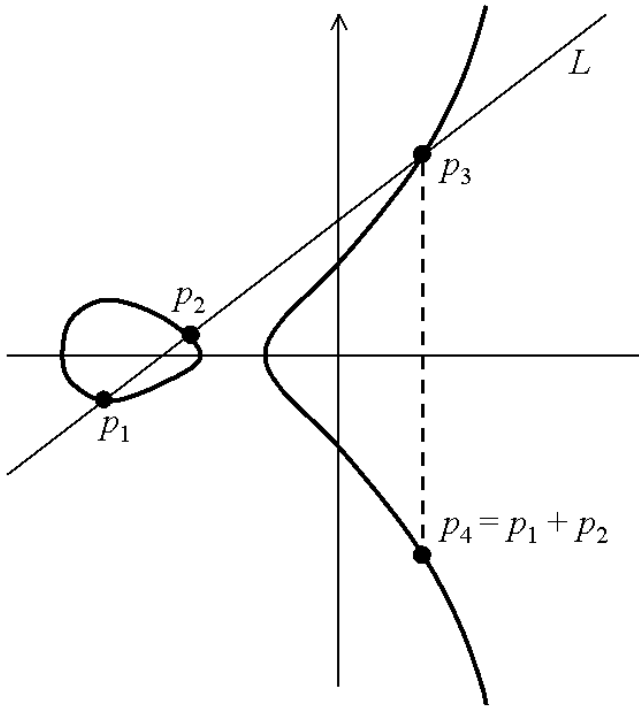
Algoritmo

1. A estrae *nonce* x
2. A calcola e invia $\tilde{m} = f(m, x)$ messaggio *blinded*
3. F calcola e invia $\tilde{s} = S(\tilde{m}, K^F_{PR}) \equiv S^B(f(m), K^F_{PR})$
4. A calcola $s = f^{-1}(\tilde{s}.\tilde{m}, x)$ firma valida

contenuto

Curve Ellittiche

- varietà abeliane
- curve algebriche piane definite da: $y^2 = x^3 + ax + b$
- non singolari: $4a^3 + 27b^2 \neq 0$



Theorem (Hasse)

sia $\mathbb{F}_q$ il campo di Galois di ordine q
sia $\mathscr{E} = \mathscr{E}_{(a,b)}(\mathbb{F}_q)$ una curva ellittica su $\mathbb{F}_q$

1. $|o(\mathscr{E}) - (q+1)| \leq 2\sqrt{q}$

$\Rightarrow$ l'ordine del campo finito usato governa la *difficoltà*

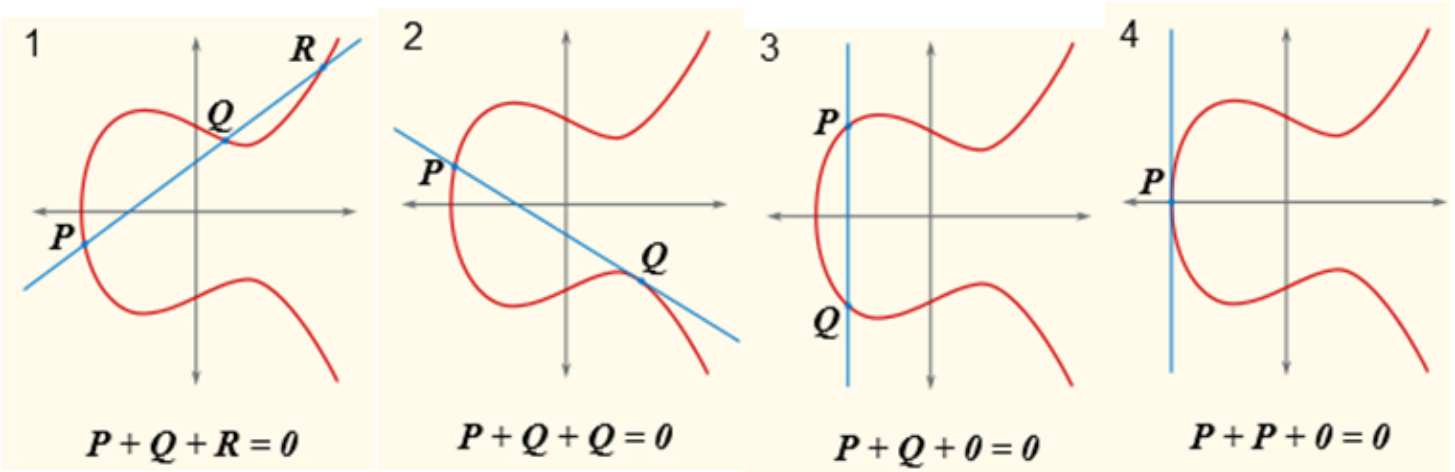
Curve Ellittiche

legge di gruppo

$$R = P + Q \triangleq (x_R, -y_R) :$$

$$\begin{cases} y_R = y_P + s(x_R - y_P) \\ x_R = \begin{cases} s^2 - 2x_P & s = \frac{3x_P^2 - p}{2y_P} \quad : x_P = x_Q \\ s^2 - x_P - x_Q & s = \frac{y_P - y_Q}{x_P - x_Q} \qquad \text{else} \end{cases} \end{cases}$$

$$R = P \times n \triangleq P + P + \ldots + P \qquad n \text{ volte}$$



Curve Ellittiche

problema matematico

$(\mathscr{E}_{(a,b)}(\mathbb{F}_q), +)$ è un gruppo abeliano

- chiusura
- associatività
- identità
- invertibilità
- commutatività

Problema: trovare $d \in [1, n-1]$, dati

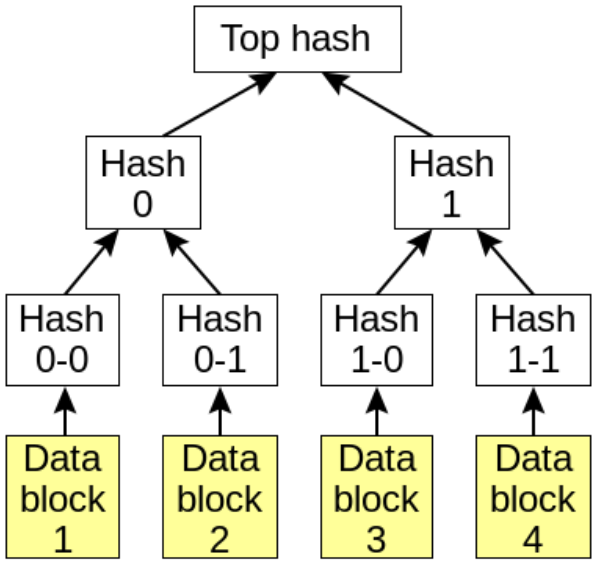
- $\mathscr{E} = \mathscr{E}_{(a,b)}(\mathbb{F}_q)$
- $G \in \mathscr{E}$: $< G > = \mathscr{E}$
- $n = o(G)$: $G \times n = O = P_\infty$, n primo
- $P \in \mathscr{E}$
- $Q = P \times d$

Funzioni di Hash

- $h : \mathbb{Z} \to \mathbb{Z}_n$ non iniettiva, *i.e.* one-way
- resistenza a
  - preimmagine $\to$ ricerca *bruta* è $O(2^n)$
  - collisioni deboli $\to$ " " "
  - collisioni forti $\to$ *birthday*: ricerca *bruta* è $O(2^{n/2}) \ll O(2^n)$
- usate soprattutto per
  - autenticazione
  - integrità
- spesso viene firmato il digest $d = h(m)$ anzichè m
  nei sistemi di autenticazione e non ripudio
- insieme ai *salt* prevengono attacchi *dizionario*

Alberi di Merkle

- foglie $\leftrightarrow$ transazioni
- altri nodi $\leftrightarrow$ hash dei loro figli
- dimostrare che una foglia $\in$ albero: $O(\log_2 N)$
- usando una lista: $O(N) \gg O(\log_2 N)$
- protegge integrità transazioni
- nati per firme *one time* di *Lamport*
- Bitcoin: $hash_n = SHA_{256}(SHA_{256}(\mathscr{L}_n|\mathscr{R}_n))$



contenuto

Transazione $\mathfrak{T}$

struttura dati

- ogni mittente appone la propria firma
- $\sum_i^N \mathbb{B}_i^{in} \geq \sum_i^M \mathbb{B}_i^{out}$

- $\mathfrak{T}^{ID} = h(\mathfrak{T})$
- timestamp t
  - $\forall i$ indirizzo di input $\mathfrak{I}_i^{in}$
    - ~~somma trasferita $\mathbb{B}_i^{in}$~~
    - chiave $K_{i,in}^{PB}$
    - indice $p : \mathfrak{I}_i^{in} = \mathfrak{I}'^{out}_p$
    - $\mathfrak{T}_{-1}^{ID} = h(\mathfrak{T}_{-1})$ precedente
    - firma $S_i(h(\mathfrak{T}_{-1}, K_{j,out}^{PB})$
  - $\forall j$ indirizzo di output $\mathfrak{I}_j^{out}$
    - somma ricevuta $\mathbb{B}_j^{out}$
    - chiave $K_{j,out}^{PB}$

Blocco $\mathfrak{B}$

struttura dati

Header

- hash del blocco $\mathfrak{B}_{i-1}$
- MerkleTree di $\{\mathfrak{T}\}_{\mathfrak{B}}$
- timestamp corrente
- target z (vedi dopo)
- nonce x
- titolare $\mathfrak{T}$ di *coinbase* (vedi dopo)

Payload

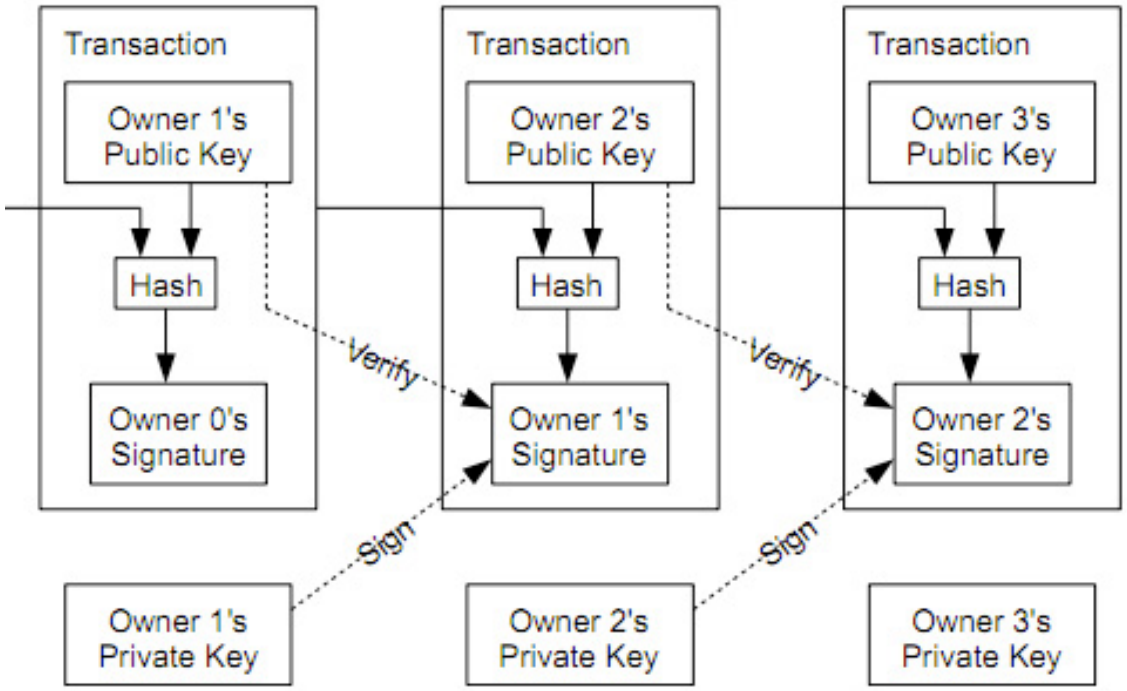
- lista transazioni $\{\mathfrak{T}\}_{\mathfrak{B}}$

- durante *mining* di $\mathfrak{B}$, campi continuamente modificati
- $\mathfrak{B}$ descrive la propria *proof of work*
  - *input*: header di $\mathfrak{B}$
  - *funzione hash*: $\mathrm{SHA}_{256}(\mathrm{SHA}_{256}(\cdot))$

Transazioni $\rightarrow$ Blocchi

generazione nuova transazione

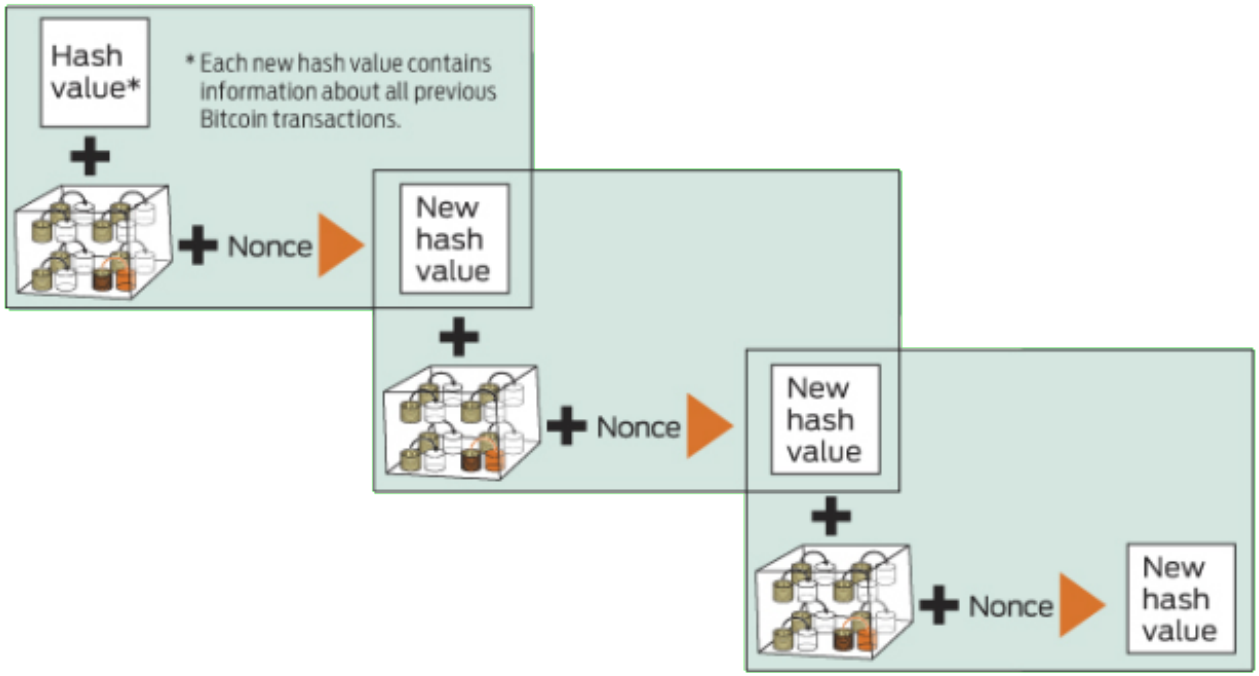
1. broadcastata nella rete tramite protocollo *flooding*
2. ogni miner **può** includerla nel suo *pool*
3. *invalida* fino a risoluzione del $\mathfrak{B}$ ove viene inserita
4. a quel punto chi l'ha ancora in pool la rimuove



Blocchi $\rightarrow$ Blockchain

generazione catena di blocchi

- hash dei blocchi precedenti $\sim$ puntatori di una lista
- a ritroso si giunge al $\mathfrak{B}$ di *genesi*



Motivazione all'utilizzo

transazioni Coinbase $\mathfrak{C}$

- $\forall \mathfrak{B},\ \exists!\, \mathfrak{C}$
- inputs: $\emptyset$
- outputs: miners che han risolto $\mathfrak{B}$
- ricompensa
  - 50 ₿ iniziali [2009]
  - dimezzata ogni 210K blocchi risolti
  - nulla dopo 6.93M blocchi [previsto 2140 A.D.]
    $\Rightarrow \sum_{i=0}^{6.93M-1} \frac{50}{2^{\lfloor i/210K \rfloor}} = 21M$ ₿
- inflazione
  - dettata solo da mining
  - limitata

Motivazione all'utilizzo

fees $\mathfrak{F}$ sulle transazioni

$\forall$ transazione $\mathfrak{T}$

- $\mathfrak{F} = \sum_i^N \mathbb{B}_i^{in} - \sum_i^M \mathbb{B}_i^{out} \geq 0$
- spetta a miners che risolvono $\mathfrak{B} \ni \mathfrak{T}$
- $\begin{cases} \textbf{mai} \text{ obbligatoria, ma...} \\ \text{miners } \textbf{mai} \text{ obbligati ad aggiungere } \mathfrak{T} \text{ a proprio pool di lavoro} \end{cases}$

$\Rightarrow$ fees incentivi per

- velocizzare validazione di $\mathfrak{T}$
- *mining* costante nonostante decrescita *coinbase rewards*

contenuto

ECDSA
inizializzazione

Alice: scelta dei parametri **pubblici**

1. $q = 2^m$
2. (a, b) : $4a^3 + 27b^2 \neq 0$
3. $G = (x_G, y_G) \in \mathscr{E}_{(a,b)}(\mathbb{F}_q)$
4. $n = o(G)$, *i.e.* $n \times G = O$

Alice: generazione coppia chiavi

1. $K^{PR} \triangleq d_A \leftarrow \text{rand} \in [1, n-1]$
2. $K^{PB} \triangleq Q_A \leftarrow n \times d_A$

Bob: verifica validità di Q_A ricevuta

1. $Q_A \neq O$
2. $Q_A \in \mathscr{E}$
3. $Q_A \times n = O$

ECDSA
firma digitale

Alice: firma del messaggio m

1. $e \leftarrow \text{hash}(m)$
2. $k \leftarrow \text{rand} \in [1, n] \subset \mathbb{N}$
3. $(x_1, y_1) \leftarrow k \times G$
4. $r \leftarrow x_1 \mod n$
5. **if** $r = 0$ **goto** 2
6. $s \leftarrow k^{-1}(e + rd) \mod n$
7. **if** $s = 0$ **goto** 2
8. **return** (r, s)

ECDSA
verifica firma

Bob: verifica firma (r, s) di m

1. $(r, s) \in [1, n-1] \times [1, n-1]$
2. $e \leftarrow \text{hash}(m)$
3. $w = s^{-1} \mod n$
4. $(u_1, u_2) \leftarrow (ew \mod n,\ rw \mod n)$
5. $(x_1, y_1) \leftarrow u_1 \times G + u_2 \times Q$
6. **ok** $\Leftrightarrow r \equiv x_1 \mod n$

SHA-256

Secure Hash Algoritm 256 bit

- $256/2 = 128$ bit di sicurezza $\Rightarrow$ collisioni non ancora trovate
- $\text{len}(m) \leq 2^{64} - 1$

1. padding: $\text{len}(m|0\ldots) \equiv 448 \mod 512$
2. padding con 64 bit di $\text{len}(m)$
3. N blocchi da 512 bit: $B^{(1)}, \ldots B^{(N)}$
4. $$H^{(i)} \triangleq H^{(i-1)} + C_{B^{(i)}}(H^{(i-1)})$$
5. ritorna $d = H^{(N)}$

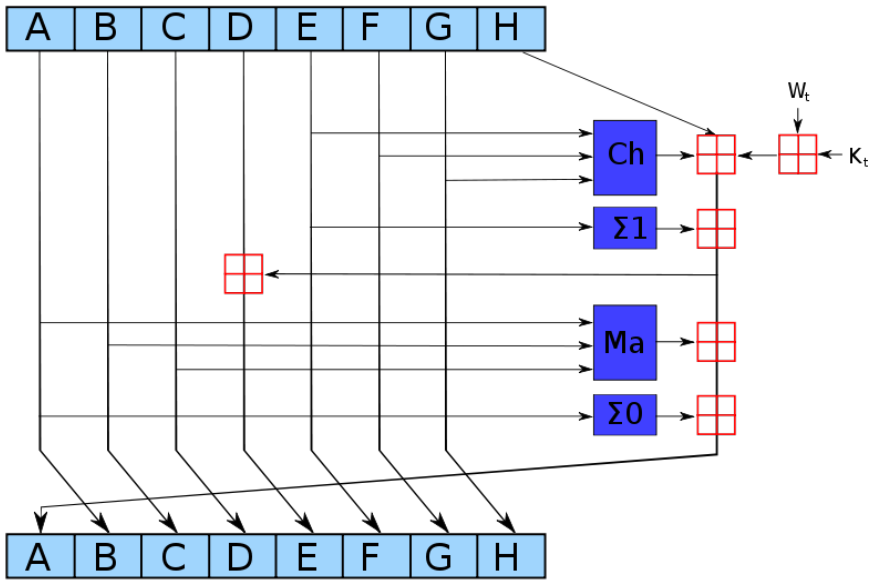


Figura: funzione di compressione C

contenuto

Firma digitale

attacco base: rottura DLOG

- $|d_{ECC}| = 256$ bit $\simeq |d_{RSA}| = 3072$ bit
- $q \simeq 10^{77}$, $n \simeq 10^{69}$, $\mathscr{E} : y^2 = x^3 + 0x + 7$ [Koblitz]

- Meet in the middle [Shank]: $\Omega(\sqrt{q})$
- nonce k dev'essere confidenziale: $d = r^{-1}(ks - e)$
- Replay attack: nonce deve tale
  a) $r_1 = r_2 = r, \quad k_1 = k_2 = k$
  b) $s_1 \equiv k^{-1}(e_1 + dr) \mod n, \quad s_2 \equiv k^{-1}(e_2 + dr) \mod n$
  c) $\ldots k(s_1 - s_2) \equiv (e_1 - e_2) \mod n$
  d) $m_1 \neq m_2 \Rightarrow (s_1 - s_2) \neq 0 \Rightarrow k \equiv (s_1 - s_2)^{-1}(e_1 - e_2) \mod n$
- malleability: $(r, s) \equiv (r, -s \mod n)$
  $\Rightarrow$ proposta: costringere $s < \frac{n}{2}$

Proof of Work

metodo Hashcash

- nato per contrastare spam, DoS: serve un'operazione onerosa
- *facile* verificare che il messaggio è soluzione di problema *difficile*
- brute force unica tecnica risolutiva
- Problema: dati $h : \mathbb{Z} \to \mathbb{Z}_n$, m, $z \leq n$, trovare nonce x :

$$d = h(m|x) < T_z = 2^{n-z+1}$$

i.e. **digest ha** z **zeri non significativi** (parametro target)

$$\Pr[d < T_z | Z = z] = \frac{1}{2^z} \Rightarrow O(2^z)$$

- problema risolto $\Leftrightarrow$ blocco $\mathfrak{B}$ risolto $\Leftrightarrow \{\mathfrak{T}\}_{\mathfrak{B}}$ convalidate

Proof of Work

exempli gratia: $z = 15$

$$\text{hash}(\text{“hello world”}|001) = 900238130012948419294712 8$$

$$\vdots \qquad \vdots$$

$$\text{hash}(\text{“hello world”}|034) = 0000834716283947104512438$$

$$\vdots \qquad \vdots$$

$$\text{hash}(\text{“hello world”}|415) = 00000000000000000000832 01$$

n.b. *gambler's fallacy*

$$\forall t_1, t_2 \quad \Pr(Z = z, T = t_1) = \Pr(Z = z, T = t_2)$$

Proof of Work

adattamento target

- target z_i ricalcolato ogni 2016 blocchi risolti $\sim$ 2 settimane
- $\Delta_i \leftarrow t_i - t_{i-1}$
- $\Delta_i \leftarrow \mathrm{clip}(\Delta_i, 0.5, 8)$
- $z_{i+1} \leftarrow z_i \frac{\Delta_i}{2}$
- $z \propto \Delta \Rightarrow$ soluzioni veloci abbassano target
  i.e. generazione problemi più difficili, vv.
- blocco risolto mediamente ogni 10 minuti

Proof of Work

sicurezza di SHA256

in teoria...

- preimage attack attack: $O(2^{256})$
- birthday attack: $O(2^{256/2})$

...in pratica

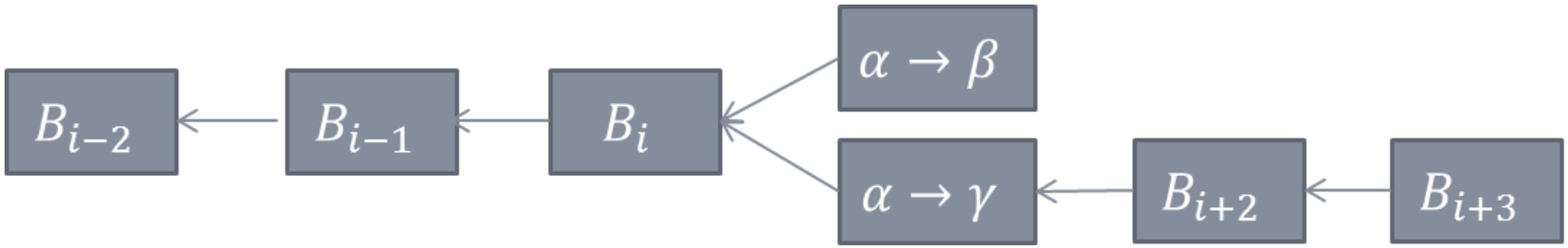
Metodo	*Attacco*	*Iterazioni*	*Complessità*
deterministico	collisione	24	$2^{28.5}$
meet in the middle	preimmagine	42	$2^{248.4}$
differenziale	pseudo collisione	46	2^{178}
biclique	preimmagine	45	$2^{255.5}$

Proof of Work

prevenzione double spending

per spendere due volte la stessa $\mathfrak{T}$ occorre

- modificarne gli output $\Rightarrow$ $\mathfrak{T}$ stessa $\Rightarrow$ $\mathfrak{B}$ di appartenenza
- ricalcolare nonce *x* per $\mathfrak{B}$ modificato
- modifica $\mathfrak{B}$ $\Rightarrow$ modifica di *N* blocchi successivi
- ricalcolare *N* nonces $\Rightarrow$ risolvere *N* problemi esponenziali



forking

- risolto imponendo aggiunta a ramo più lungo
- sotto ipotesi *web of trust* sopravviverà il ramo corretto

Web of trust

dato un pool di miners $\mathfrak{M}$ con capacità di calcolo $\mathscr{C}_{\mathfrak{M}}$ $[\mathrm{GH/s}]$

$\Pr[\mathfrak{M} \text{ risolva blocco}] \propto \frac{\mathscr{C}_{\mathfrak{M}}}{\mathscr{C}_{\Omega}}$

$\Rightarrow$ Bitcoin funziona solo se $\mathscr{C}_{\Theta} \geq 50\% \, \mathscr{C}_{\Omega}$

se $\exists \, \mathfrak{M}_{\overline{\Theta}}$ pool disonesto: $\mathscr{C}_{\mathfrak{M}_{\overline{\Theta}}} \geq 50\% \, \mathscr{C}_{\Omega}$

$\Rightarrow$ catena più lunga comanderebbe $\Rightarrow$ crollo fiducia $\Rightarrow$ crollo valore

- no motivazione di lucro diretto
- ma no difesa contro nemici abbastanza potenti e motivati

contenuto

attacco alla firma digitale

unico modo per forgiare ₿: usare quelli di qualcuno

- il problema di forgiare dal nulla **non ha senso**
- occorre appropriarsi di K_{PR} $\Rightarrow$ rompere ECDSA

se quantum computers implementati

- rottura ECDSA può diventare *facile*
- inversione SHA_{256} resta *difficile*
- indirizzi $\mathfrak{I} = \mathrm{SHA}_{256}(\mathrm{SHA}_{256}(K_{PB}))$
  - ottenere K_{PB} dal solo $\mathfrak{I}$ è *difficile*
  - ma se è nota il sistema è rotto

caso di studio: Mt.GoX [2014]

prima del crollo

exchange

- web service di scambio valute *fiat* con ₿
- 2011: compromissione account interno al perimetro $\Rightarrow$ furto ₿
- 2013: Blockchain fork in rami con regole diverse
  $\Rightarrow$ MtGox sospende transazioni

malleabilità

- $\mathfrak{T}^{ID} \triangleq h(\mathfrak{T}), \quad S \in \mathfrak{T}$
- codice *MtGox* non controlla la validità del formato di S
- problema $\in$ implementazione, $\notin$ protocollo ₿
- $\exists$ altri sistemi migliori per proprio storico $\Rightarrow$ immunità

caso di studio: Mt.GoX [2014]

il crollo: attacco delle transazioni *mutanti*

frode di *Eve* ai danni di *MtGox*

1. *M* invia $\mathfrak{T}$ a *E* dopo legittima richiesta prelievo
2. *E* ritocca $S(\mathfrak{T})$ prima della sua conferma
3. ora $\exists \tilde{\mathfrak{T}} \equiv \mathfrak{T}$, ma $h(\tilde{\mathfrak{T}}) \neq h(\mathfrak{T}) \Rightarrow \tilde{\mathfrak{T}}^{ID} \neq \mathfrak{T}^{ID}$
4. $\tilde{\mathfrak{T}}$ diffusa da *E* e confermata prima di $\mathfrak{T}$
5. $\mathfrak{T}$ non confermata perchè $\tilde{\mathfrak{T}}$ invalido
6. *E* manda un complain a *M* per $\mathfrak{T}$ non ricevuta
7. *M* controlla il suo storico: $\mathfrak{T}$ non è stata accettata
8. *M* costretto a inviare $\mathfrak{T}'$ come rimborso

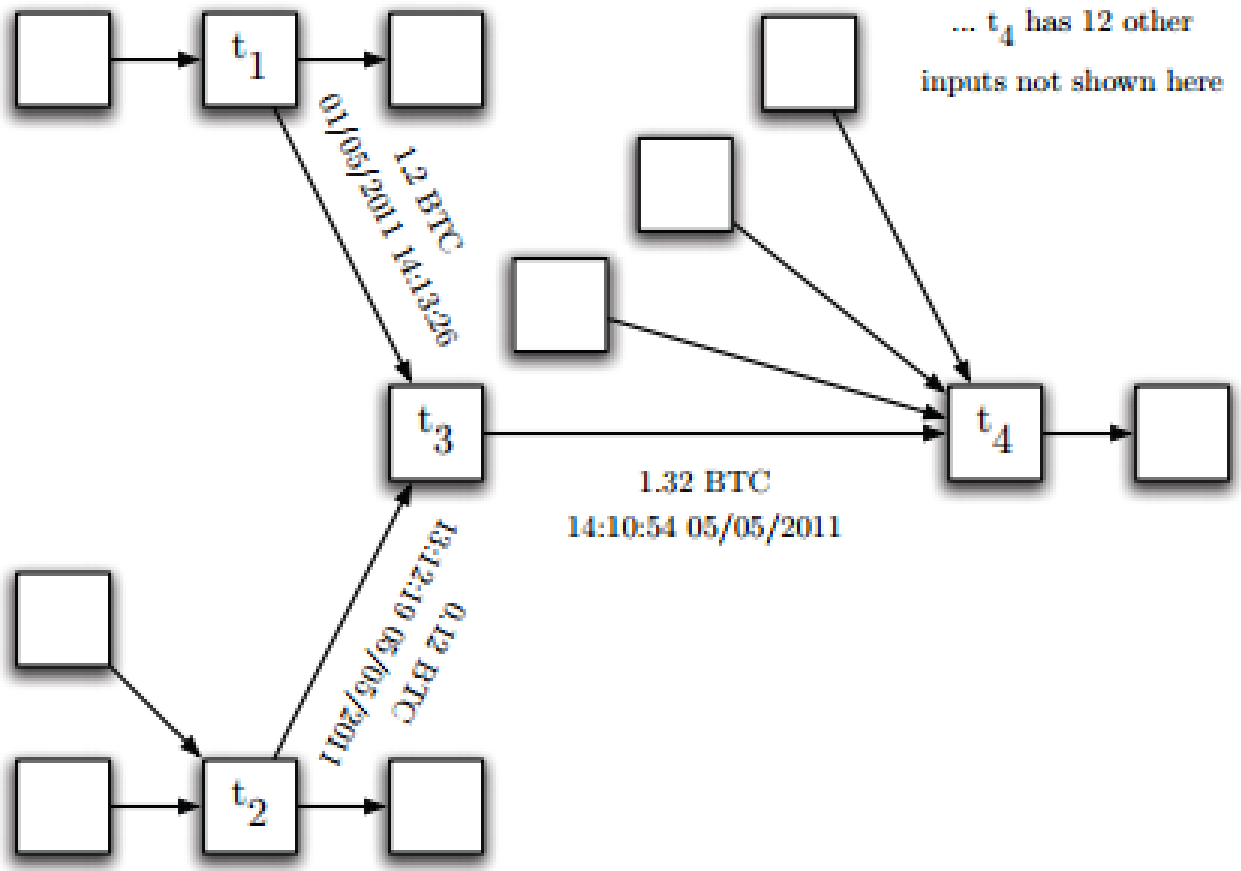
DDoS

1. *M* riceve tante transazioni mutanti
2. grande sovraccarico, persino se c'è resistenza a frode
3. latenza nelle risposte $\Rightarrow$ incertezza $\Rightarrow$ speculazione

contenuto

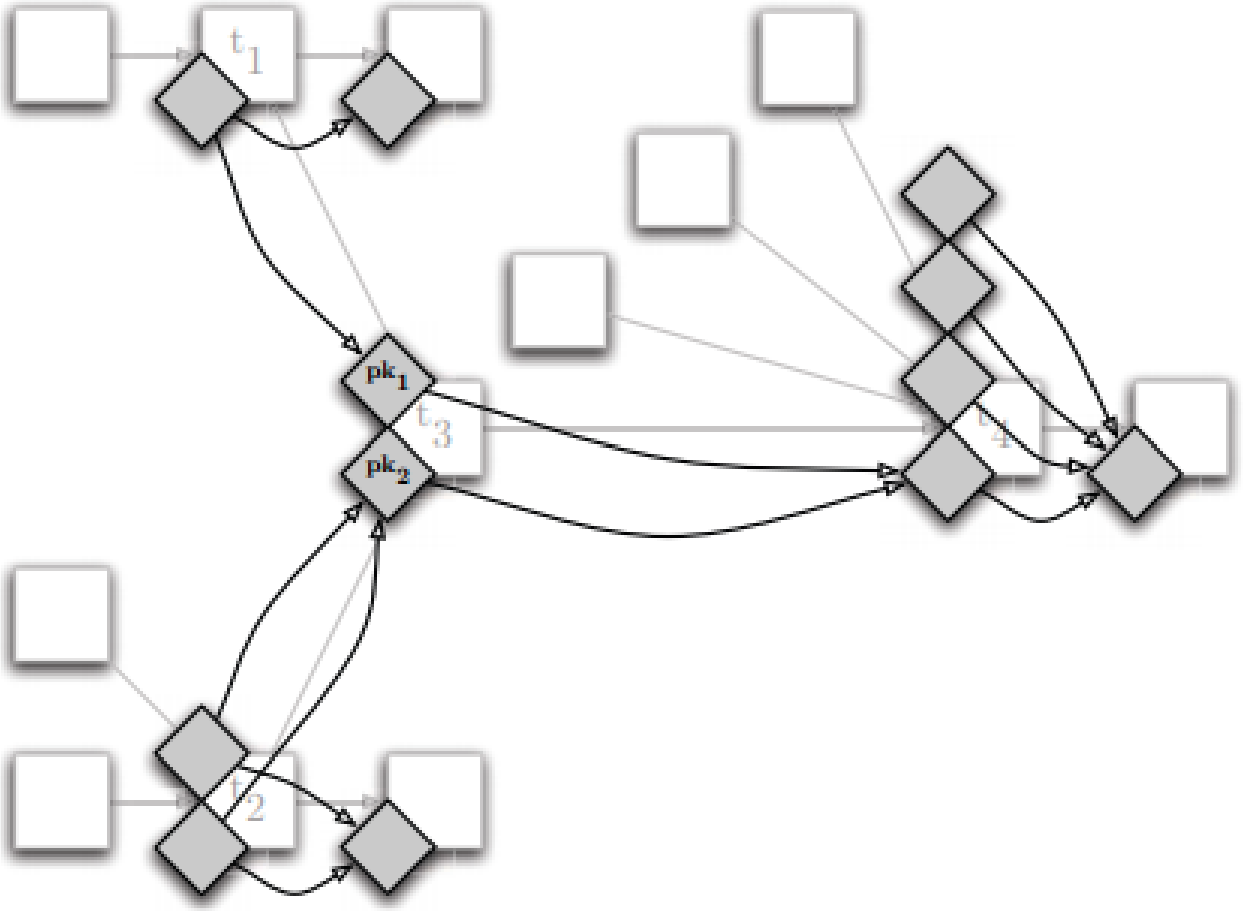
caso di studio: Reid [2011]

rete transazioni $\mathscr{T}$



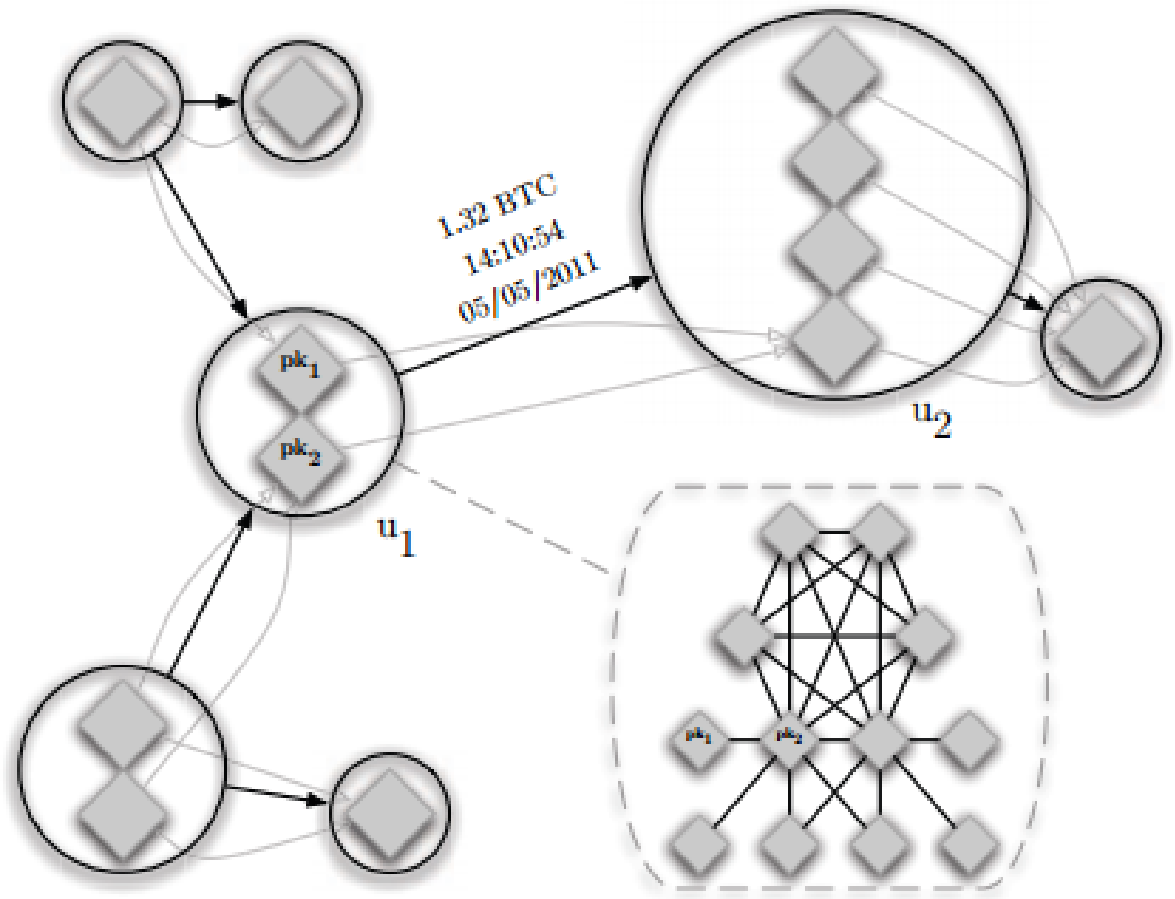
caso di studio: Reid [2011]

rete utenti imperfetta $\tilde{\mathscr{U}}$



caso di studio: Reid [2011]

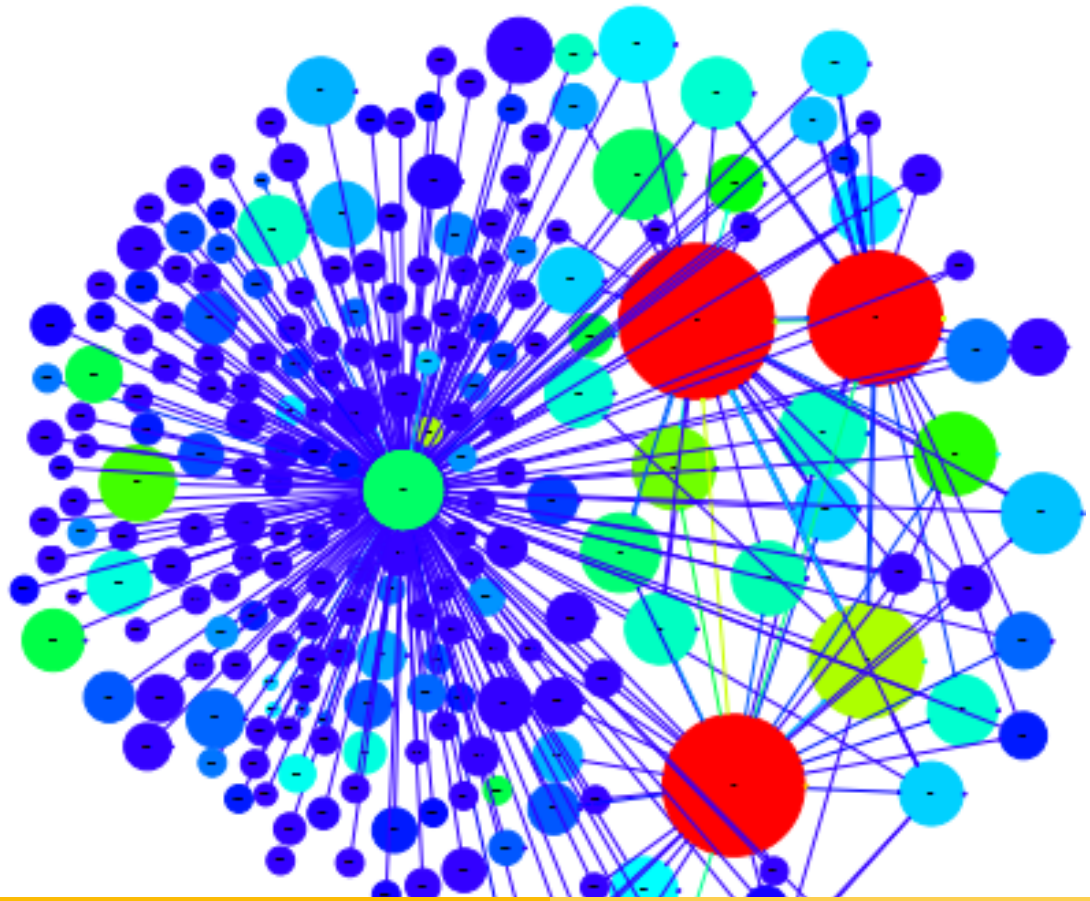
rete utenti $\mathscr{U}$, rete ancella $\mathscr{A}$



caso di studio: Reid [2011]

integrazione con informazioni esterne

- dimensione $\propto |\{K_{PB}\}|$ utente
- colore $\propto$ ₿ scambiati



contenuto

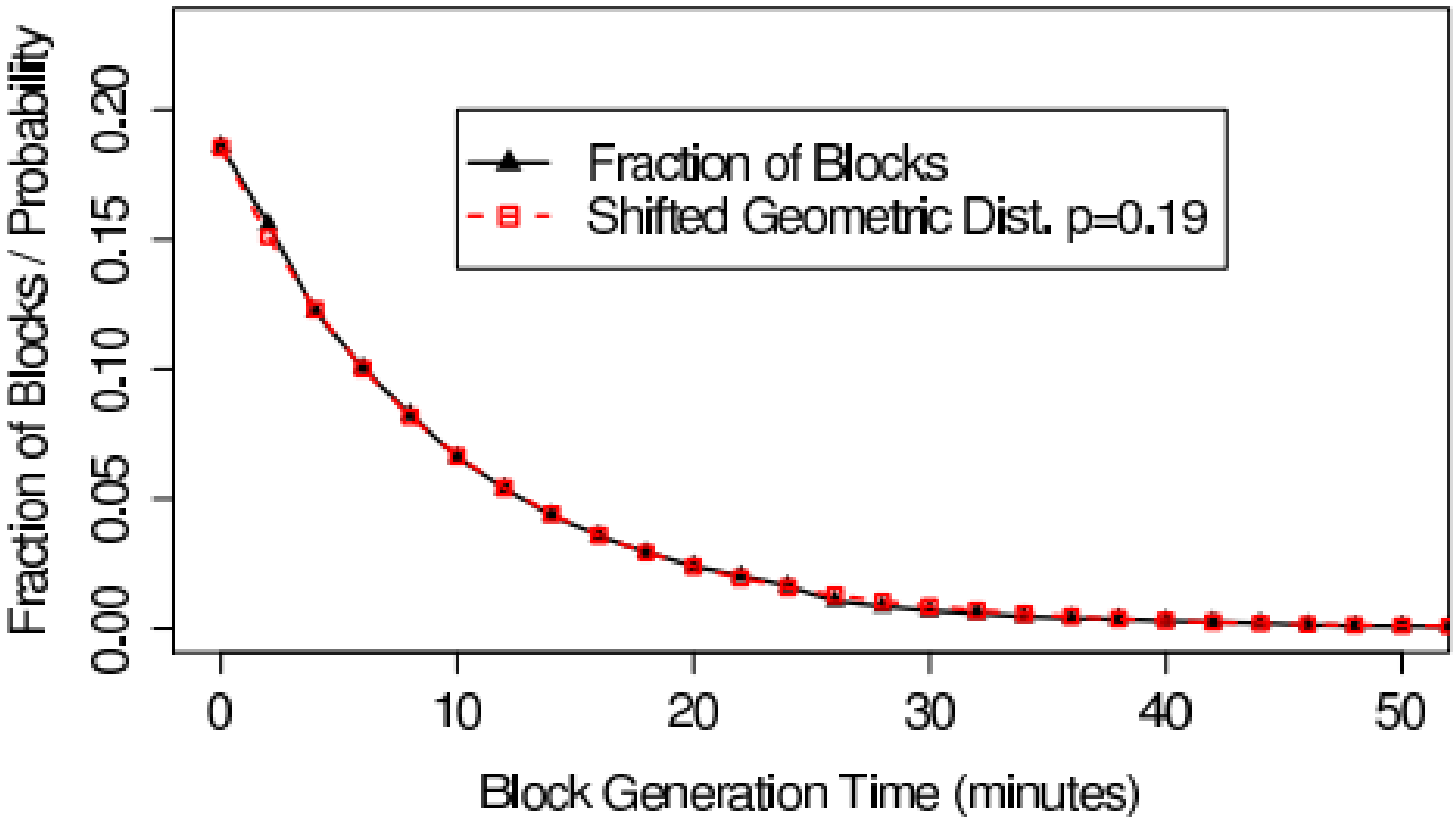
caso di studio: Karame [2012]

tipologia transazione

- *lenta, e.g.* acquisto ticket eventi
  $\Rightarrow$ sicurezza offerta dal mining
- *veloce, e.g.* pagamento in negozio
  $\Rightarrow$ Karame: $\exists$ possibilità di double spending
  - tempi scambio [s] $\ll$ tempi validazione [min]
  - Bitcoin segue tecnica dello struzzo
  - problema limitato in gravità ma non risolto

caso di studio: Karame [2012]

garantire validità nel caso *veloce*



caso di studio: Karame [2012]
ipotesi

hosts

- A peer disonesto
- H complici di A
- V vendor onesto

transazioni

- $\mathfrak{T}_V$: acquisto regolare
- $\mathfrak{T}_A$: recupero fraudolento

ipotesi

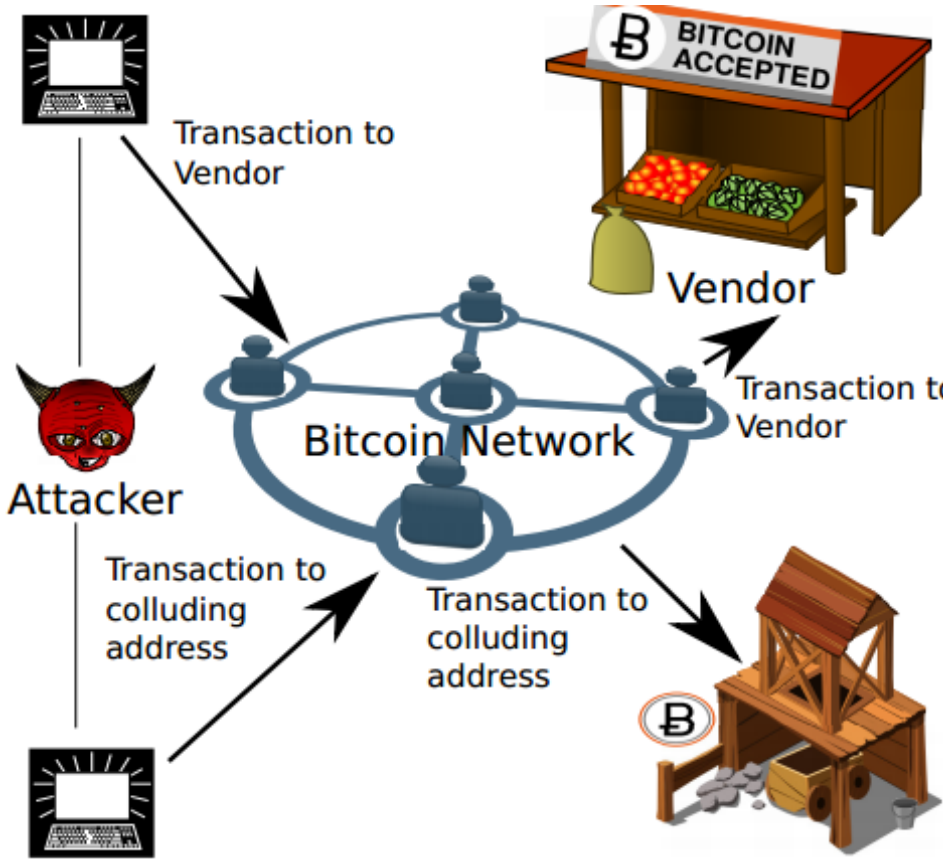
- A conosce indirizzo IP di V
- capacità di calcolo di A trascurabile
- $\mathfrak{I}_V^{in} = \mathfrak{I}_A^{in} \in A$
- $V \ni \mathfrak{I}_V^{out} \neq \mathfrak{I}_A^{out} \in A$
- implementazioni clients: *plain vanilla*

caso di studio: Karame [2012]

idea di massima

- $\mathfrak{T}_V, \mathfrak{T}_A$ inviate contemporaneamente
  $\Rightarrow$ incluse nello stesso pool
- se $\mathfrak{T}, \mathfrak{T}'$ condividono inputs
  $\Rightarrow$ non ammesse nello stesso pool
- inclusa solo la prima $\mathfrak{T}$ ad arrivare

  $\Rightarrow$
- $\mathfrak{T}_A$ da validare rapidamente
- $\mathfrak{T}_V$ sarà smentita dalla rete



caso di studio: Karame [2012]

1ª **condizione**: connessione diretta

V riceve prima $\mathfrak{T}_V$ di $\mathfrak{T}_A$ oppure V includerebbe prima $\mathfrak{T}_A$ nel pool

- A conosce IP di V
- client accetta sempre nuove connessioni < 125 max
- A comunica con H
  - senza latenza
  - privatamente
- H non comunica con V
- A invia
  1. $\mathfrak{T}_V$ a V
  2. $\mathfrak{T}_A$ a H

$\Rightarrow t_V^V < t_V^A$

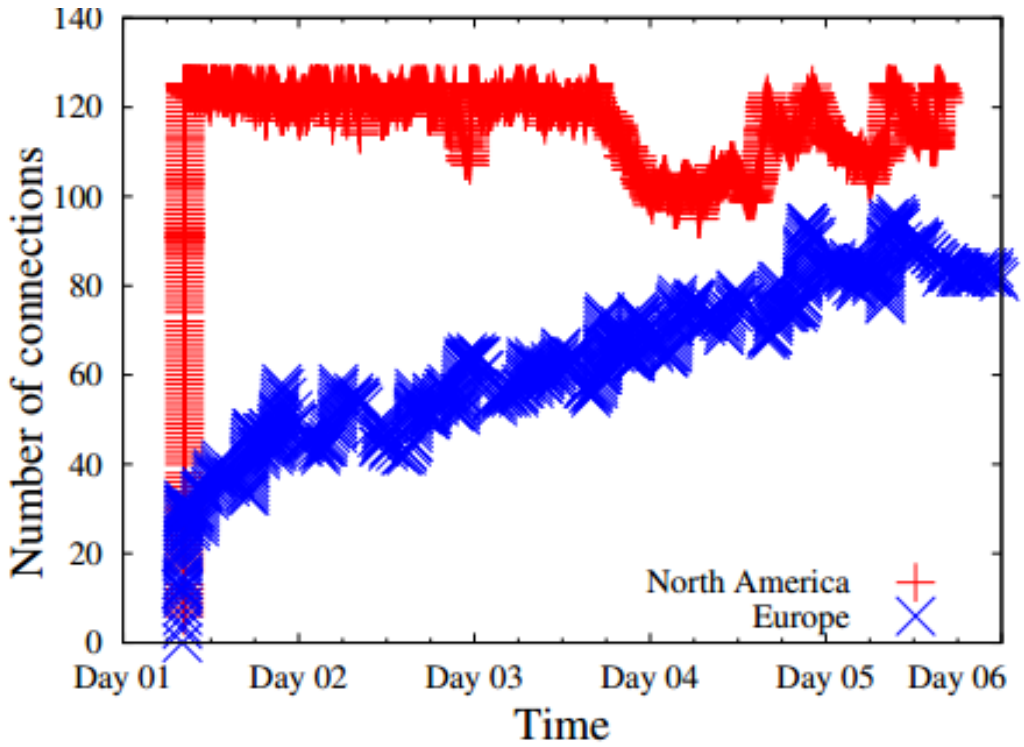


Figura: analisi del momento propizio

caso di studio: Karame [2012]

2ª condizione: diffusione manipolata

$\mathfrak{T}_A$ inclusa in blockchain (*i.e.* confermata) prima di $\mathfrak{T}_V$

oppure $\mathfrak{T}_A$ non potrebbe seguire $\mathfrak{T}_V$ in blockchain

- H, V probabilmente lontani $\Rightarrow \mathfrak{T}_A, \mathfrak{T}_V$ broadcastate finchè
  $\begin{cases} \text{ogni peer include } \mathfrak{T}_A \dot{\vee} \mathfrak{T}_V \text{ in proprio pool} \\ \mathfrak{T}_A \vee \mathfrak{T}_V \text{ è confermata} \end{cases}$
- $\Pr[\tau_A < \tau_V] \propto \eta_A / \eta_V$ governabile in due modi
  - invio di $\mathfrak{T}_A$ precede invio di $\mathfrak{T}_V$
  - H aiutano A diffondendo $\mathfrak{T}_A$ e filtrando $\mathfrak{T}_V$
- ulteriori ipotesi
  - $\exists$ istante ove $\mathfrak{T}_A$, $\mathfrak{T}_V$ convivono
  - $\forall \varepsilon$ p.a.p., $\Pr[\tau_A \sim \text{secs}] \cup \Pr[\tau_V \sim \text{secs}] < \varepsilon$
  - η_A, η_V non scambiano blocchi risolti $\Rightarrow \tau_A$, τ_V indipendenti

caso di studio: Karame [2012]

probabilità di successo

Pr[successo in un tempo δt] $\sim$ Bernoulli(η, p)

- η = numero di peers coinvolti
- p = Pr[peer generi un $\mathfrak{B}$ in δt]

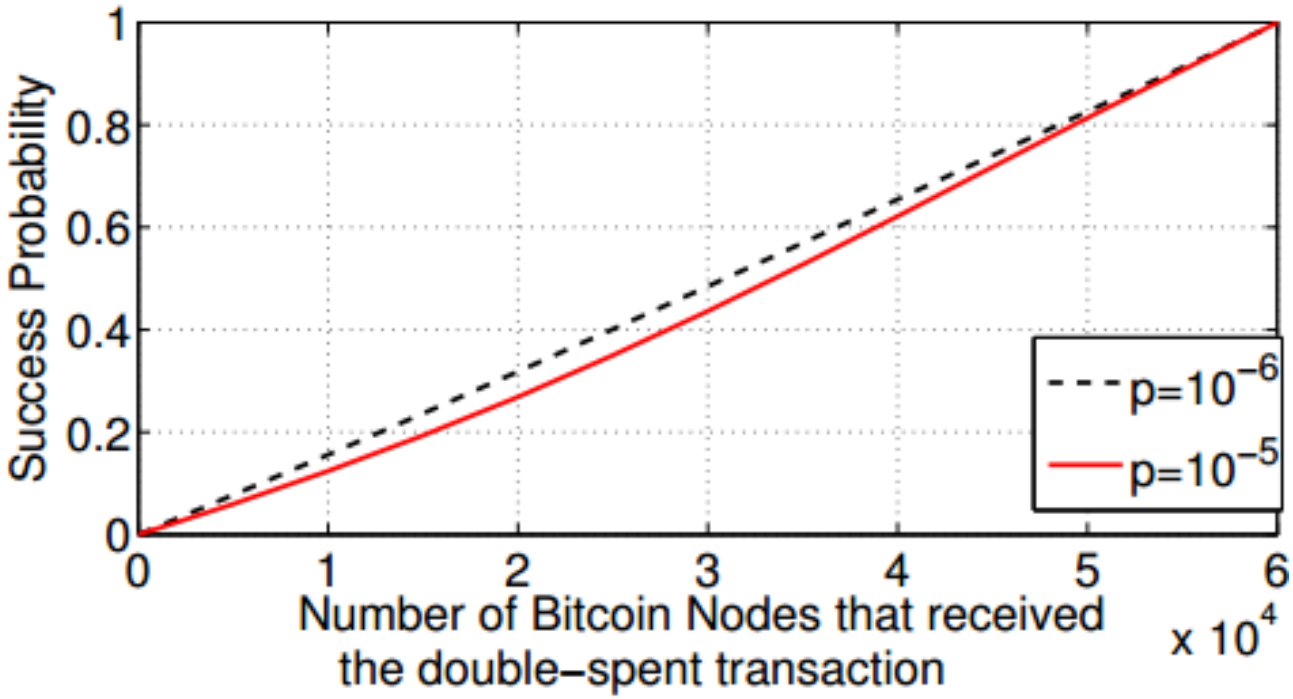


Figura: Pr[successo | $\delta t = 10s$, $\eta = 6 \cdot 10^4$]