introduzione a Bitcoin

e alle criptovalute digitali

P.Bertoni

28 aprile 2014

Tabella Contenuti

contenuto

Criptovaluta
specifiche sul protocollo

problema

implementare una valuta economica

- affidabile
- informatica
- decentralizzata
- distribuita

Criptovaluta

specifiche sulle transazioni

problema

trasmettere transazioni di valuta tra enti

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

registrate in una sorta di storico globale

Criptovaluta

idea astratta di transazione $\mathfrak{T}$

- atto tra N mittenti e M destinatari
- utenti incoraggiati a usare un indirizzo unico $\forall\, \mathfrak{T}$
- $\sim$ assegno
  "in data t, $\{x\}_1^N$ *ha versato* tot *a* $\{y\}_1^M$ *che ora ne è proprietario"*

gestione del resto

- $y' \in \{y\}$ destinatari, controllato da chi emette $\mathfrak{T}$
- deframmentare: nuova $\mathfrak{T}$ con molteplici mittenti

Criptovaluta

specifiche sull'affidabilità: **proof of work**

problema

progettare algoritmo di mining per convalida $\mathfrak{T}$

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

motivazione a partecipare

- onestà ricompensata
- complessità lavoro onesto $\equiv$ complessità disonesto
- Pr[successo disonestà] $\rightarrow 0$

Differenze con valute tradizionali

ontologia

esplicita: unità fisiche | implicita: in funzione di $\mathfrak{T}$

---

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 autore e F 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 invia $\overline{m} = f(m, x)$ messaggio *cieco* a F
3. F restituisce $s^B = \overline{s} = E(\overline{m}, K^F_{PR})$
4. chiunque conosca K^F_{PB} calcola $\overline{m}^S = E(\overline{s}, K^F_{PB})$
5. " " pure x " $m^S = f^{-1}(\overline{m}^S, x)$

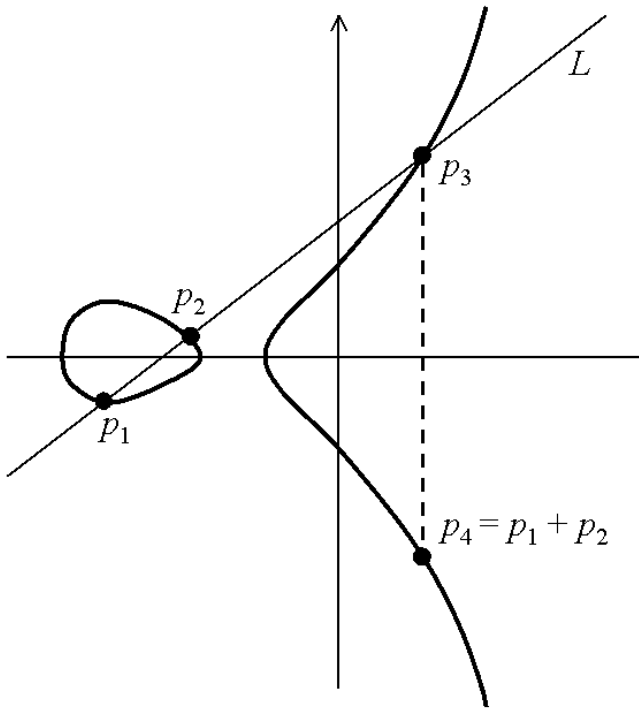
contenuto

Curve Ellittiche

- curve definite su un certo $\mathbb{F}_q$ da

$$y^2 = x^3 + ax + b$$

- non singolari, *i.e.* $4a^3 + 27b^2 \neq 0$



Theorem (Hasse)

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

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

$\Rightarrow$ ordine *GF* governa *difficoltà*

Curve Ellittiche

legge di gruppo: definizione

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

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

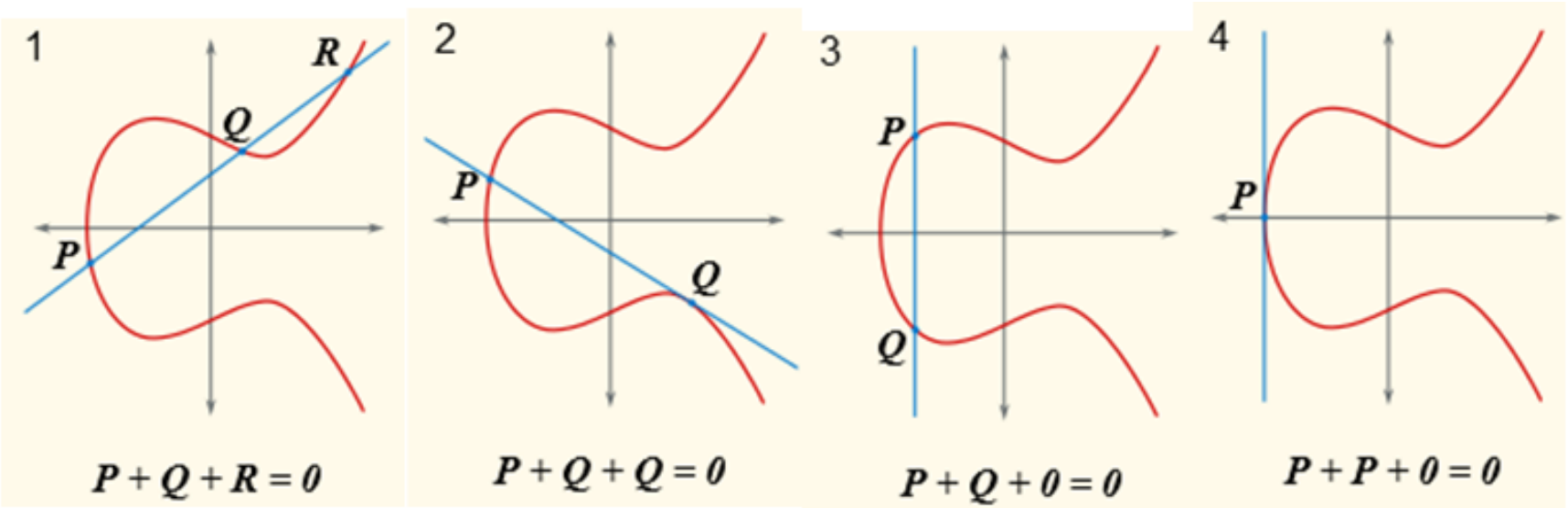
$$x_P \neq x_Q \ : \ \begin{cases} y_R \triangleq y_P + s(x_R - y_P) \\ x_R \triangleq s^2 - x_P - x_Q \end{cases} \qquad s = \frac{y_P - y_Q}{x_P - x_Q}$$

$$x_P = x_Q \ : \ \begin{cases} y_P = -y_Q \ : & R = O \\ y_P = y_Q \neq 0 \ : & \begin{cases} y_R \triangleq y_P + s(x_R - y_P) \\ x_R \triangleq s^2 - 2x_P \end{cases} \qquad s = \frac{3x_P^2 + a}{2y_P} \end{cases}$$

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

Curve Ellittiche

legge di gruppo: casistica



Curve Ellittiche

problema matematico

trovare un segreto $d \in [1, n-1]$, 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
- $P \in \mathscr{E}$
- $Q = P \times d$

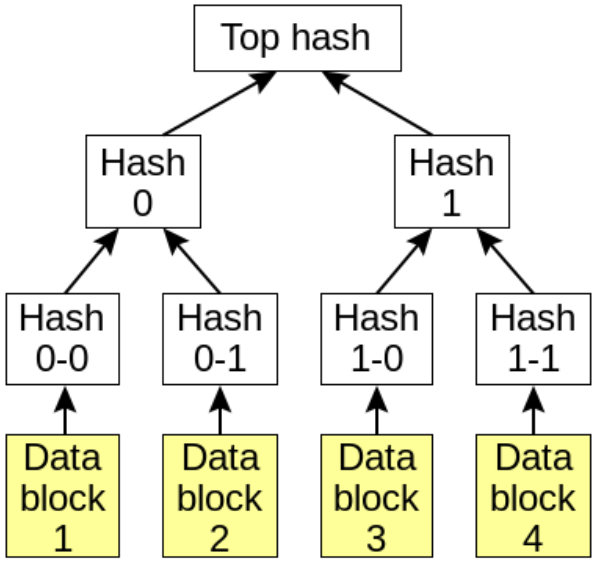
Funzioni di Hash One Way

- $h : \mathbb{Z} \to \mathbb{Z}_n$ non iniettiva
- 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 firmato il digest $h(m)$ anzichè m

Alberi di Merkle

usati in protezione integrità transazioni

- foglia $\leftrightarrow \mathfrak{T}$
- $\neg$ foglia $\leftrightarrow$ hash dei due figli
- decidere se una foglia $\in$
  - lista: $O(N)$
  - albero: $O(\log_2 N) \ll O(N)$
- Bitcoin: $\mathrm{hash}(n) = SHA_{256}(SHA_{256}(\mathscr{L}_n|\mathscr{R}_n))$



contenuto

Transazione $\mathfrak{T}$

struttura dati

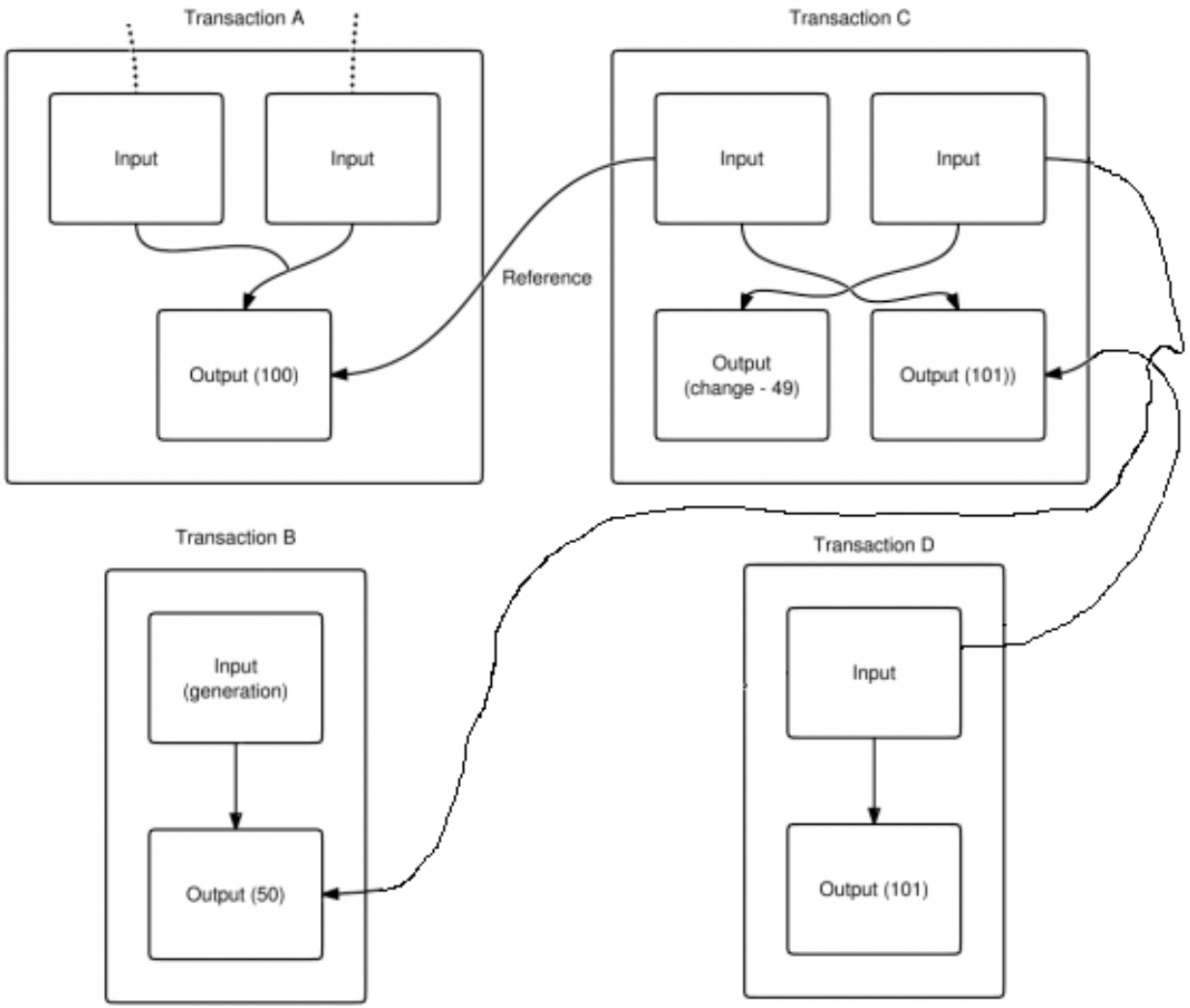
- $\mathfrak{T}^{ID} = \text{hash}(\mathfrak{T})$
- timestamp t

- $\forall i$ indirizzo di input $\mathfrak{I}_i^{in}$
  - ~~somma trasferita ฿_i^{in}~~
  - chiave $K_{i,in}^{PB}$
  - indice $p : \mathfrak{I}_i^{in} = [\mathfrak{I}_p^{out}]_{\mathfrak{T}_{-1}}$
  - $\mathfrak{T}_{-1}^{ID} = \text{hash}(\mathfrak{T}_{-1})$
  - firma $\mathfrak{T}^{S_i} = E(\tilde{\mathfrak{T}}_i, K_{i,in}^{PR})$

- $\forall j$ indirizzo di output $\mathfrak{I}_j^{out}$
  - somma ricevuta ฿_j^{out}
  - hash$(K_{j,out}^{PB})$

Transazione $\mathfrak{T}$

collegamento tra transazioni



Blocco $\mathfrak{B}$

struttura dati

Header

- hash di $\mathfrak{B}_{i-1}^{\mathfrak{H}}$
- MerkleTree di $\{\mathfrak{T}\}_{\mathfrak{B}}$
- timestamp t
- target z
- nonce x
- titolare della *coinbase* $\mathfrak{C}$

Payload

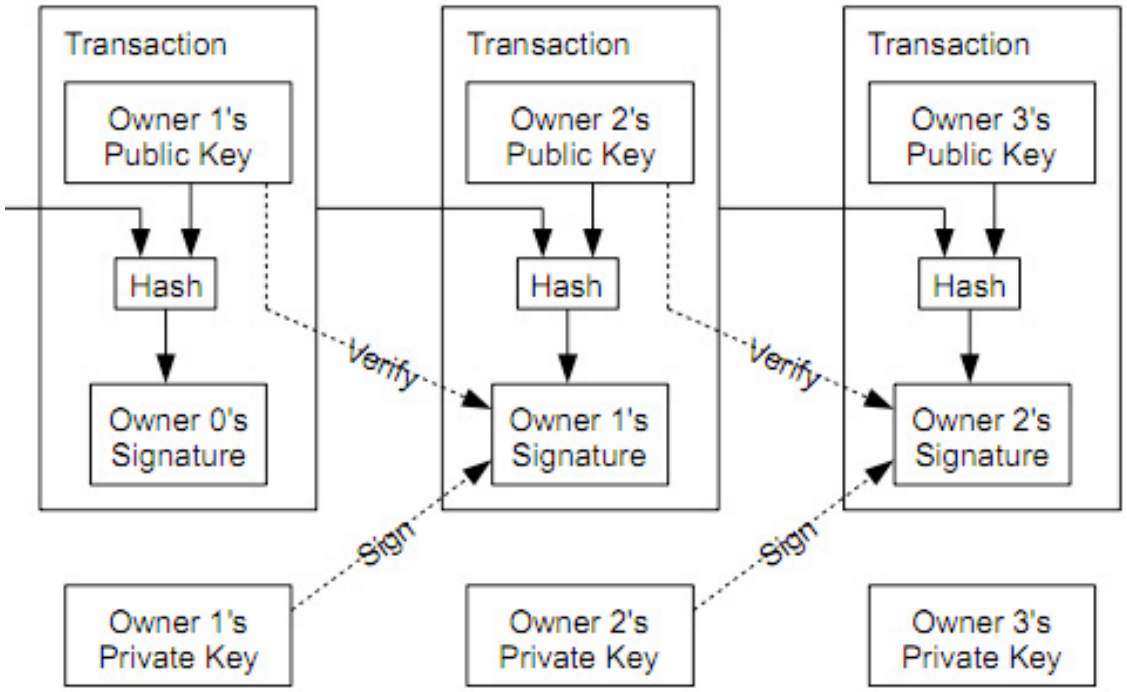
- lista transazioni $\{\mathfrak{T}\}_{\mathfrak{B}}$

- durante *mining* di $\mathfrak{B}$, campi continuamente modificati
- $\mathfrak{B}$ descrive la propria *proof of work*

Transazioni $\rightarrow$ Blocchi

generazione nuova transazione

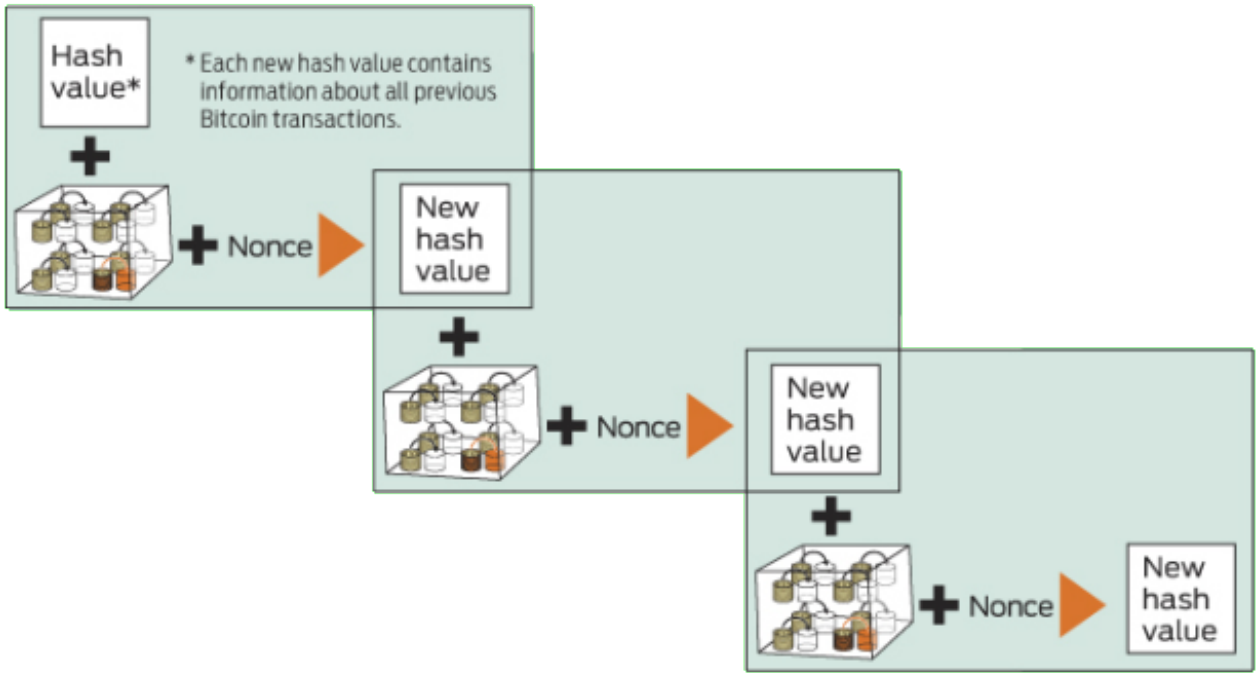
1. broadcastata tramite protocollo *flooding*
2. ogni miner **può** includerla nel suo *pool*
3. inizialmente inserita in un pool come *invalida*
4. dopo risoluzione del $\mathfrak{B}$ corrente è rimossa da ogni pool



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

fees $\mathfrak{F}$ sulle transazioni

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

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

incentivi per

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

Motivazione all'utilizzo

transazioni Coinbase $\mathfrak{C}$

- $\forall \mathfrak{B}, \exists! \mathfrak{C}$
- inputs: $\emptyset$
- outputs: ricompensa a miners risolutori di $\mathfrak{B}$
  - *coinbase*
    - 50 ₿ iniziali
    - dimezzata ogni 210K blocchi risolti
    - nulla dopo 6.93M blocchi
  - $\{\mathfrak{F}\} \in \mathfrak{B}$
- inflazione
  - dettata solo da mining
  - limitata

$$\sum_{i=0}^{6.93M-1} \frac{50}{2^{\lfloor i/210K \rfloor}} = 21M \text{ ₿}$$

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. $n = o(G)$

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 collisioni non ancora trovate
- $\text{len}(m) < 2^{64}$, $\text{len}(n) = 256$

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

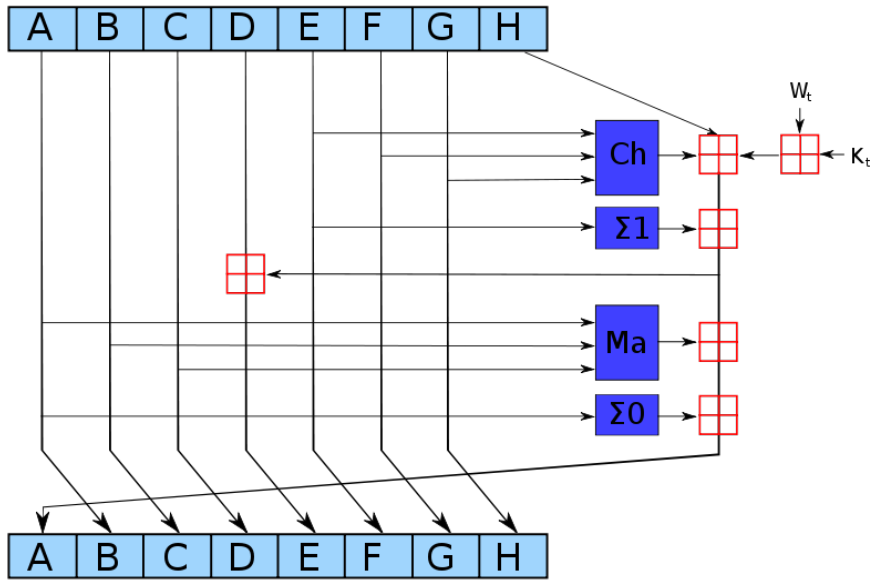


Figura: funzione di compressione C

contenuto

Firma digitale

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

- meet in the middle [Shank]: $\Omega(\sqrt{q})$
- nonce k è confidenziale: $d = r^{-1}(ks - e)$
- replay attack: nonce deve tale
  a) $r_1 = r_2 = r$
  b) $s_1 \equiv k^{-1}(e_1 + dr) \mod n$, $s_2 \equiv k^{-1}(e_2 + dr) \mod n$
  c) $k(s_1 - s_2) \equiv (e_1 - e_2) \mod n$
  d) $m_1 \neq m_2 \Rightarrow (s_1 - s_2) \neq 0 \Rightarrow k \equiv (s_1 - s_2)^{-1}(e_1 - e_2) \mod n$

Proof of Work

metodo Hashcash

- *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 :

$$d = \mathrm{hash}^{\mathrm{PoW}}(m|x) < T_z = 2^{n-z+1}$$

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
- Bitcoin: $\mathrm{hash}^{\mathrm{PoW}}(\mathfrak{B}) = SHA_{256}(SHA_{256}([\mathrm{Header}]_{\mathfrak{B}}))$

Proof of Work

exempli gratia: $z = 15$

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

$$\vdots \qquad \vdots$$

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

$$\vdots \qquad \vdots$$

$$\text{hash}(\text{“hello world”}|415) = 00000000000000000000 83201$$

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

1. $\Delta_i^t \leftarrow t_i - t_{i-1}$
2. $\Delta_i^t \leftarrow \mathrm{clip}(\Delta_i, 0.5, 8)$
3. $z_{i+1} \leftarrow z_i \frac{\Delta_i^t}{2}$

- $z \propto \Delta^t \Rightarrow$ 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: $O(2^{256})$
- birthday attack: $O(2^{256/2})$

...in pratica

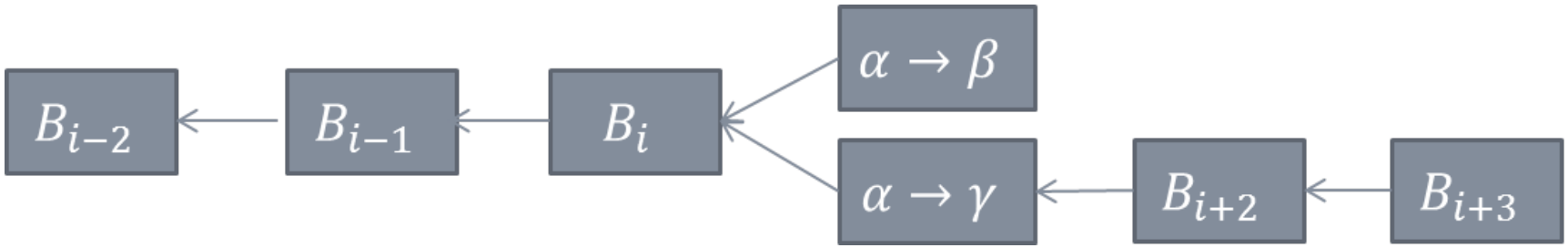
Metodo	*Attacco*	*Iterazioni*	*Complessità*
deterministico	collisione	24	$2^{28.5}$
meet in the middle	preimmagine	42	$2^{248.4}$
differenziale	pseudo collisione	46	2^{178}
biclique	preimmagine	45	$2^{255.5}$

Proof of Work

prevenzione double spending

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

1. modifica $\mathfrak{I}^{out} \Rightarrow \mathfrak{T}$ stessa $\Rightarrow \mathfrak{B}$ di appartenenza
   ricalcolo nonce x
2. modifica $\mathfrak{B} \Rightarrow$ modifica N blocchi successivi
   ricalcolo N nonces $\Rightarrow$ risolvere N problemi esponenziali



forking

- *policy*: aggiungere sempre a ramo più lungo
- in una *web of trust* sopravviverà il ramo corretto

Web of trust

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

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

$\Rightarrow$ condizione necessaria Bitcoin: $\mathscr{C}_{\text{fair}} \geq 50\%\ \mathscr{C}_{\Omega}$

se $\exists\ \mathfrak{M}_{\text{unfair}}$ pool disonesto *t.c.* $\mathscr{C}_{\mathfrak{M}_{\text{unfair}}} \geq 50\%\ \mathscr{C}_{\Omega}$

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

- no motivazione diretta di lucro
- ma problema irrisolvibile perchè sistematico

contenuto

forgiatura

unica possibilità: rubarli a qualcuno

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

se quantum computers implementati

- rottura ECDSA *può* diventare *facile*
- collisione 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 sistema rotto

caso di studio: MtGoX [2014]

prima del crollo

web service di exchange

- scambio valute *fiat* con ₿
- 2013: Blockchain fork in rami con regole diverse
  $\Rightarrow$ *MtGox* sospende transazioni

malleabilità

- $\mathfrak{T}^{ID} \triangleq h(\mathfrak{T}), \quad \mathfrak{T}^{S} \in \mathfrak{T} \quad \Rightarrow \quad \mathfrak{T}^{ID}$ dipende dalla sua firma
- codice *MtGox* accetta firme malformate
- problema $\in$ implementazione, $\notin$ protocollo ₿
- $\exists$ altri exchanges più rigorosi $\Rightarrow$ immunità

caso di studio: MtGoX [2014]

il crollo: attacco delle transazioni *mutanti*

frode di *Eve* ai danni di *MtGox*

1. *M* invia $\mathfrak{T}$ a *E* come prelievo
2. *E* ritocca $\mathfrak{T}^S$ 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*, confermata prima di $\mathfrak{T}$
5. $\mathfrak{T}$ non confermata perchè $\tilde{\mathfrak{T}}$ invalido
6. *E* fa 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 rimborso $\mathfrak{T}'$

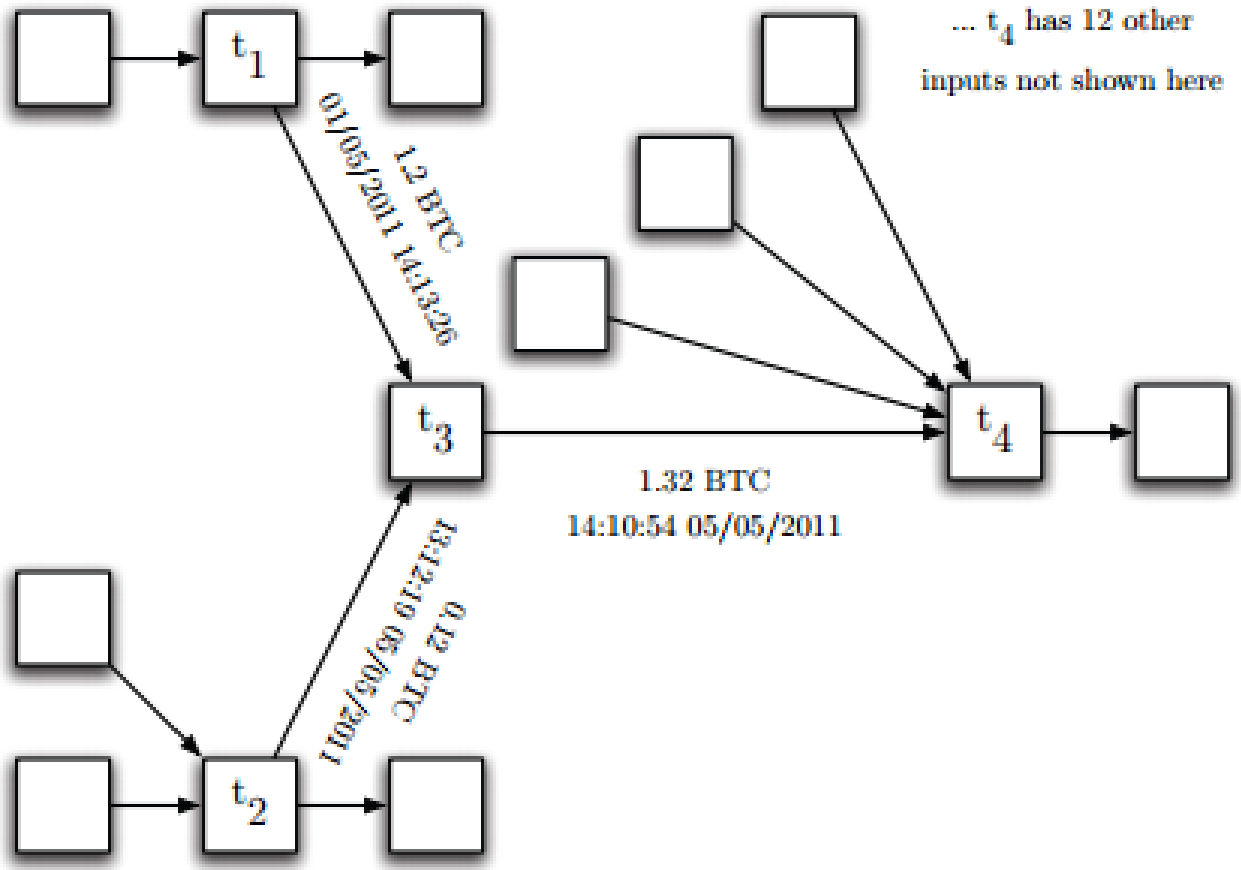
DDoS

- grande sovraccarico, persino se c'è immunità
- latenza nelle risposte $\Rightarrow$ incertezza $\Rightarrow$ speculazione

contenuto

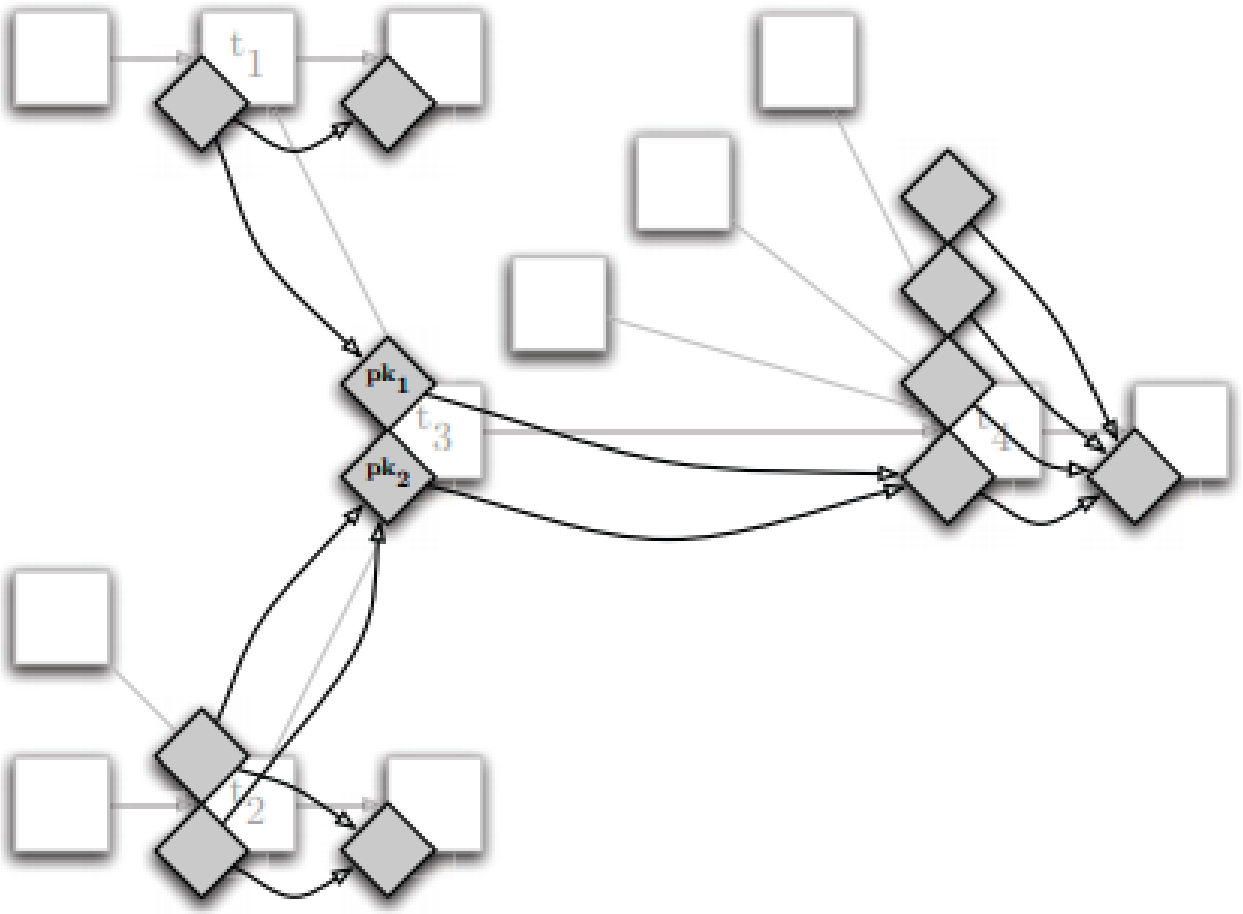
caso di studio: Reid [2011]

rete transazioni $\mathscr{T}$



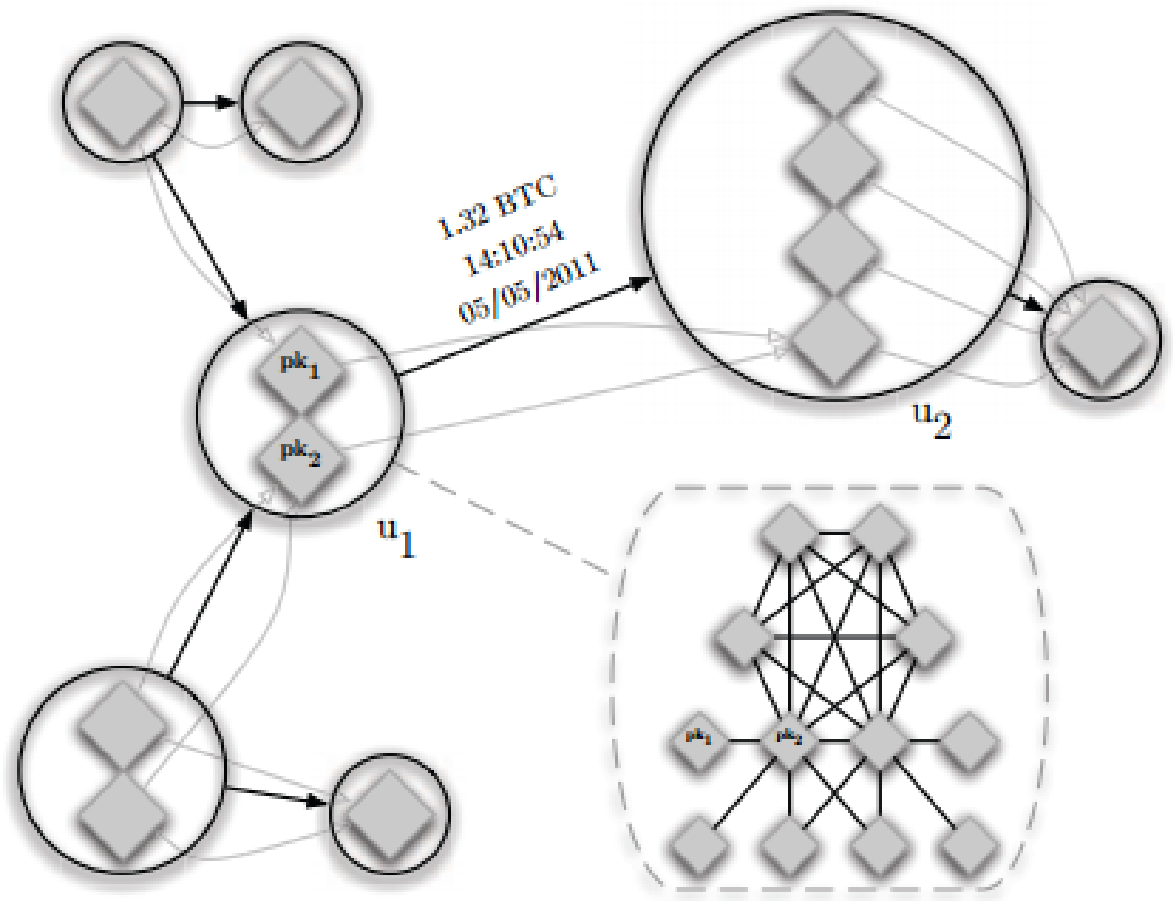
caso di studio: Reid [2011]

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



caso di studio: Reid [2011]

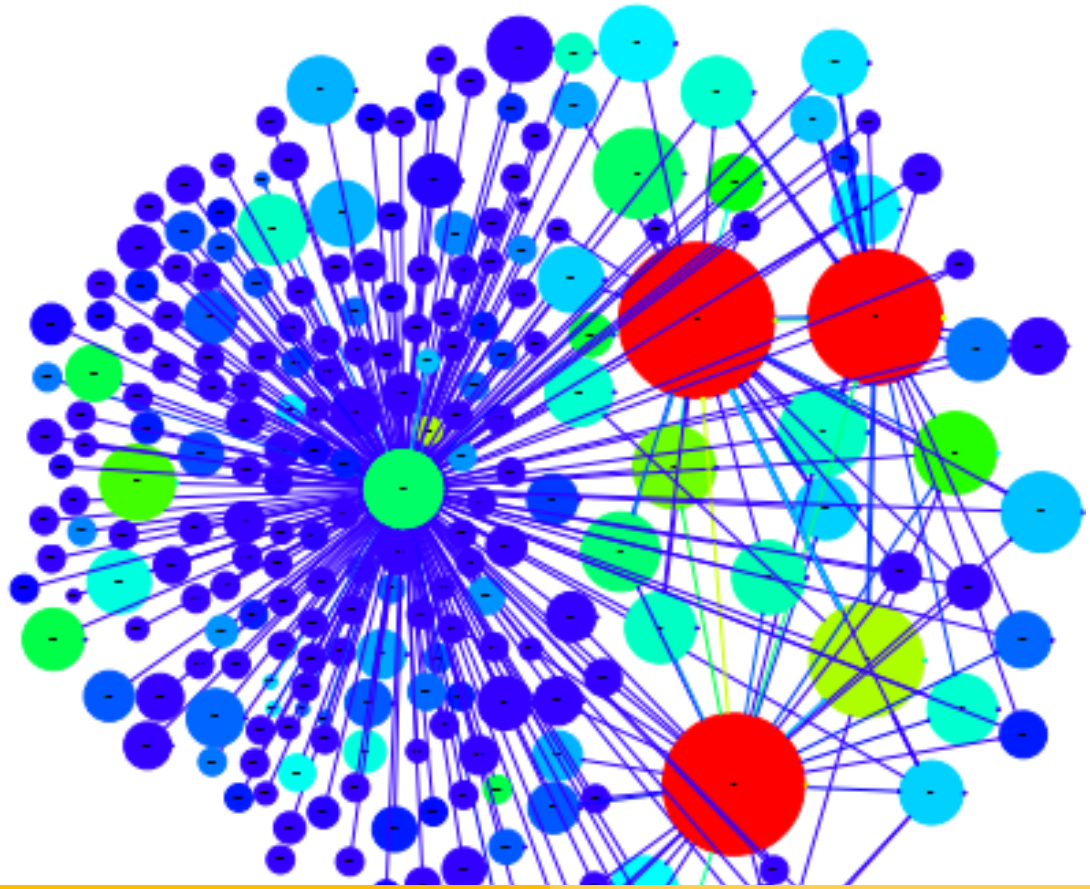
rete utenti $\mathscr{U}$, rete ancella $\mathscr{A}$



caso di studio: Reid [2011]

integrazione con informazioni esterne

- dimensione $\propto |\{K_{PB}\}|$ utente = # transazioni
- colore $\propto$ ฿ scambiati



contenuto

caso di studio: Karame [2012]

tipologia transazione

- lenta, *e.g.* acquisto ticket eventi

  sicurezza offerta dal mining

- veloce, *e.g.* pagamento in negozio

  $\exists$ possibilità di *double spending*

  - tempi scambio [*s*] $\ll$ tempi validazione [*min*]
  - Bitcoin segue tecnica *struzzo*
  - problema non grave ma aperto

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

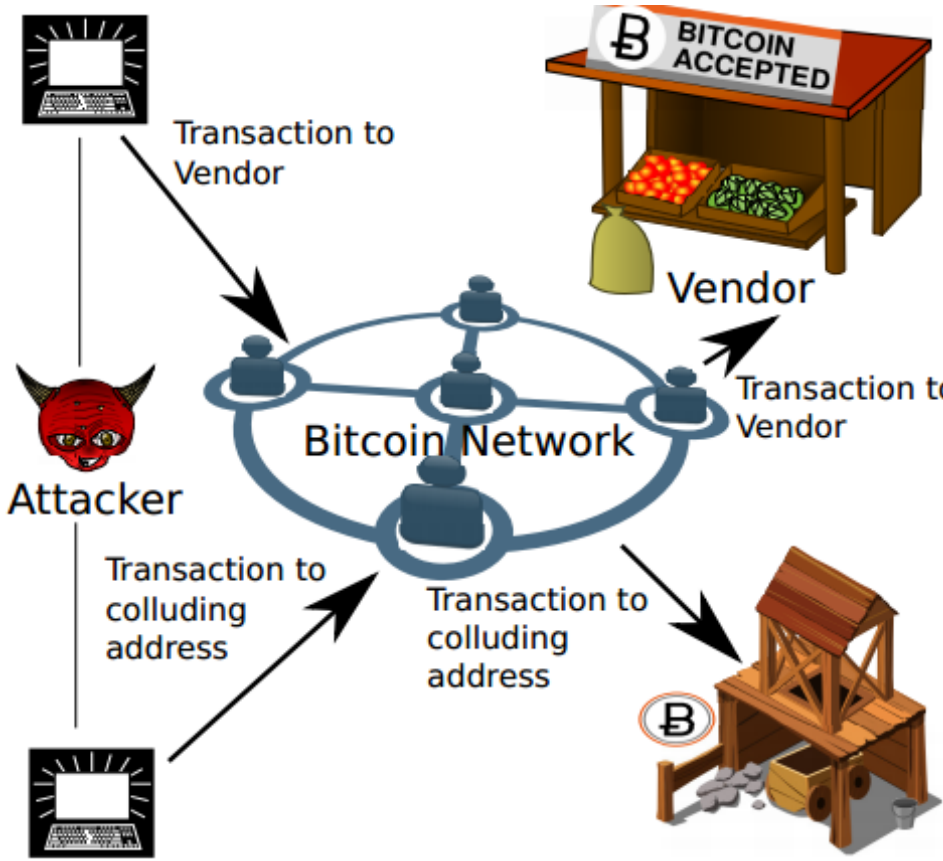
ipotesi

- A conosce indirizzo IP di V
- $\mathscr{C}_A$ 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 *plain vanilla*

caso di studio: Karame [2012]

idea di massima

- $\mathfrak{T}_V, \mathfrak{T}_A$ inviate contemporaneamente
  $\Rightarrow$ incluse nello stesso pool
- se $\mathfrak{I}^{in}_{\mathfrak{T}} = \mathfrak{I}^{in}_{\mathfrak{T}'}$
  $\Rightarrow$ 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]

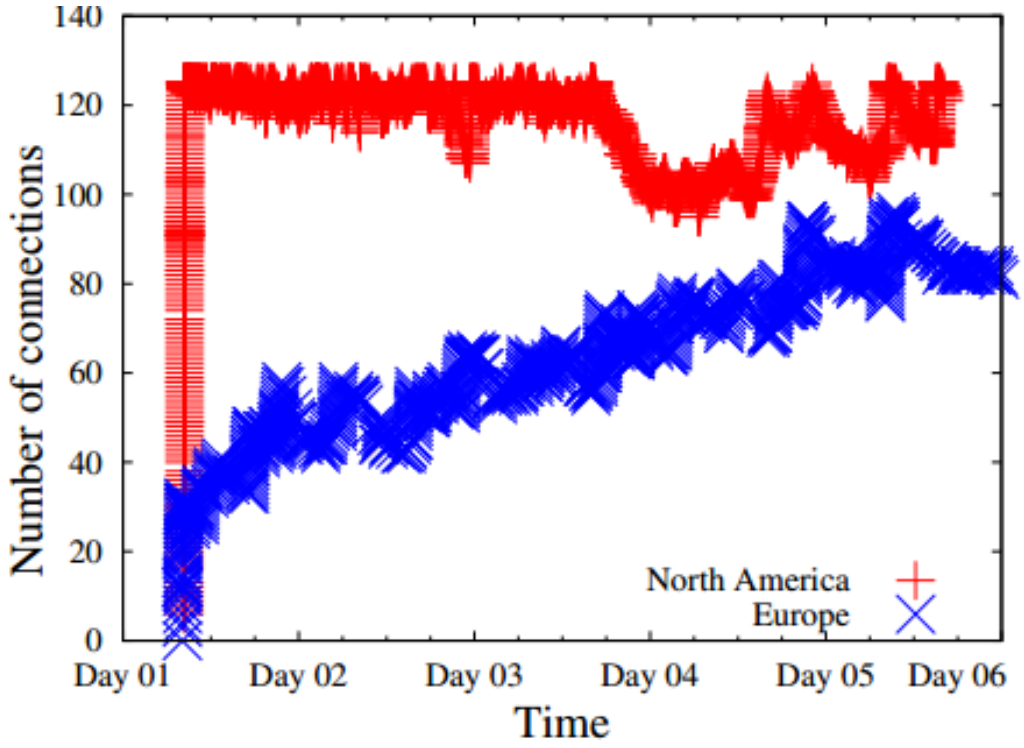
1ª condizione: connessione diretta tra A e V

V riceve prima $\mathfrak{T}_V$ di $\mathfrak{T}_A$

oppure V includerebbe prima $\mathfrak{T}_A$ nel suo pool

- 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

$$\Rightarrow t_V^V < t_V^A$$



caso di studio: Karame [2012]

2ª condizione: diffusione manipolata

$\mathfrak{T}_A$ confermata in *blockchain* prima di $\mathfrak{T}_V$

oppure $\mathfrak{T}_A$ non più validabile

- ogni peer include $\mathfrak{T}_A \dot{\vee} \mathfrak{T}_V$ in proprio pool
  - $\mathfrak{T}_A, \mathfrak{T}_V$ broadcastate in due partizioni
  - termine quando $\mathfrak{T}_A \dot{\vee} \mathfrak{T}_V$ confermata
- $\Pr[\tau_A < \tau_V] \propto \eta_A / \eta_V$ migliora se
  - invio di $\mathfrak{T}_A$ precede invio di $\mathfrak{T}_V$
  - *H* aiutano *A* diffondendo $\mathfrak{T}_A$ e filtrando $\mathfrak{T}_V$

caso di studio: Karame [2012]

probabilità di successo

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

- η_A = # peers coinvolti
- p = Pr[peer generi $\mathfrak{B}$ in δt]

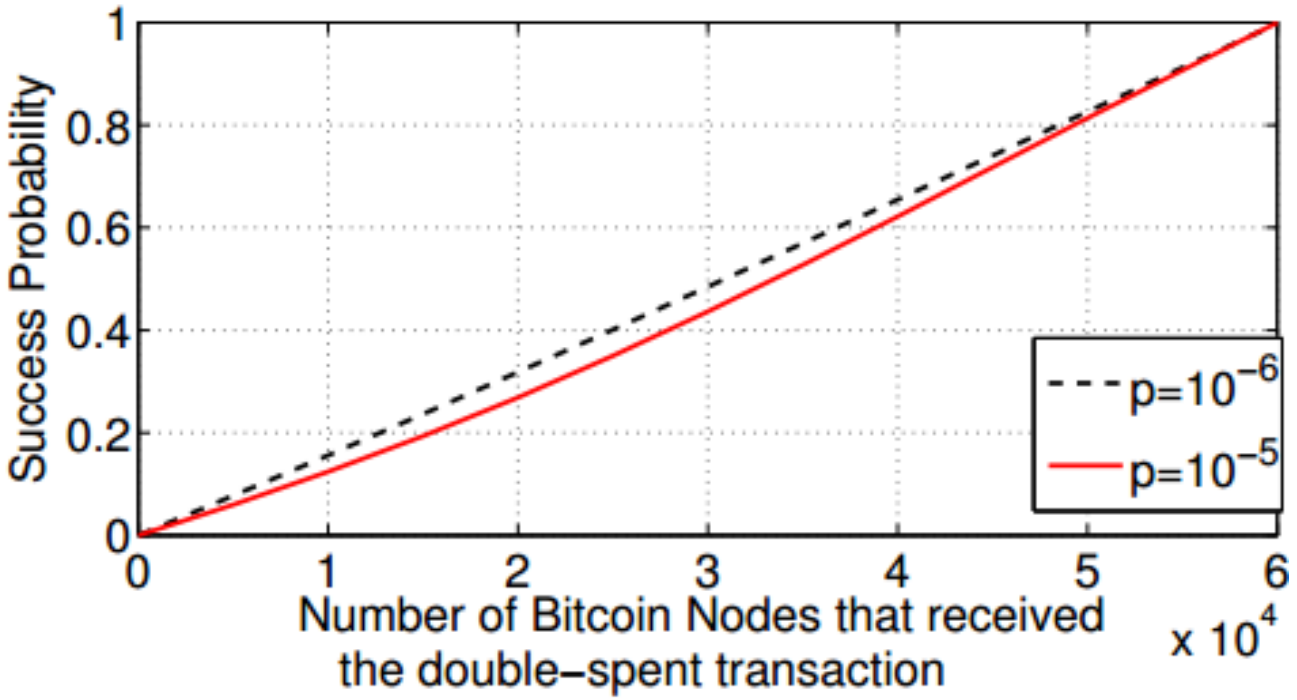


Figura: Pr[successo | $\delta t = 10s$, $\eta = 6 \cdot 10^4$]