Ergodic Mean-Payoff Games for the Analysis of Attacks in Crypto-Currencies

Krishnendu Chatterjee[1] Amir Kafshdar Goharshady[1]
Rasmus Ibsen-Jensen[1] Yaron Velner[2]

[1]IST Austria

[2]Hebrew University of Jerusalem

CONCUR 2018

Outline

Outline

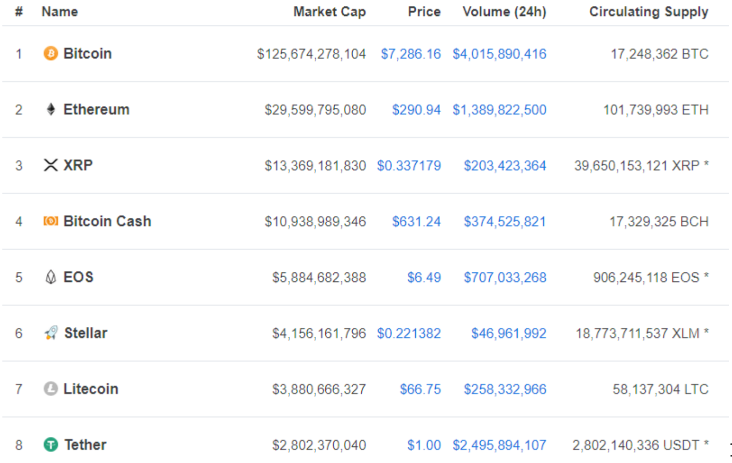
Quantitative Analysis of Security Violations

- Automated security analysis of programs is usually qualitative
- It uses qualitative properties, e.g. safety or liveness, to ensure absolute security
- but absolute security is sometimes impossible or too costly
- In these cases, we want to quantify and limit the costs of attacks $\rightarrow$ Quantitative Analysis
- What does cost mean? Is cost always well-defined?
- For Cryptocurrency protocols, it is.

Outline

Cryptocurrencies

- It all started with Bitcoin, but nowadays there are thousands of cryptocurrencies out there.



#	Name	Market Cap	Price	Volume (24h)	Circulating Supply
1	Bitcoin	$125,674,278,104	$7,286.16	$4,015,890,416	17,248,362 BTC
2	Ethereum	$29,599,795,080	$290.94	$1,389,822,500	101,739,993 ETH
3	XRP	$13,369,181,830	$0.337179	$203,423,364	39,650,153,121 XRP *
4	Bitcoin Cash	$10,938,989,346	$631.24	$374,525,821	17,329,325 BCH
5	EOS	$5,884,682,388	$6.49	$707,033,268	906,245,118 EOS *
6	Stellar	$4,156,161,796	$0.221382	$46,961,992	18,773,711,537 XLM *
7	Litecoin	$3,880,666,327	$66.75	$258,332,966	58,137,304 LTC
8	Tether	$2,802,370,040	$1.00	$2,495,894,107	2,802,140,336 USDT *

[1]

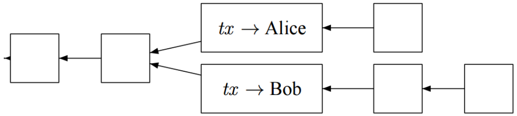
[1] coinmarketcap.com

Cryptocurrencies

- No outside governance, no central bank
- Everything works based on the Blockchain decentralized consensus protocol
- The protocol assumes that a majority of the network is honest
- It only dictates the outcomes of actions, but not the actions themselves
- The whole ecosystem is game-theoretic
- Transactions are irreversible
- It's a safety-critical system
- We need Formal Quantitative Analysis

Double Spending
The most basic attack

- Peer-to-peer transfer is not safe, because one can simply copy the coins
- So, let's announce all the transfers to the whole network
- Still not safe

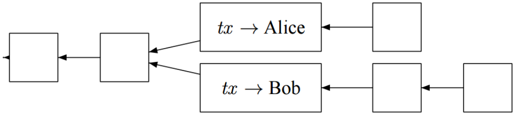


- Bitcoin's solution: Blockchain and Mining

Blockchain

and Mining

- Transactions are grouped into blocks
- There is a distributed ledger of blocks, called the Blockchain
- Every node in the network keeps a local copy of the Blockchain
- Mining: in order to add a block, one must solve a hard computational puzzle
- In Bitcoin the puzzle is to invert a hash function, $f(\text{previous block}, \text{current block}, \text{miner's id}, \text{nonce}) < c$



- The longest chain is the consensus chain

Incentives for Mining

- Transaction Fees
- Block Rewards (currently 12.5 BTC)
  - This is how new units of currency are formed

Pool Mining

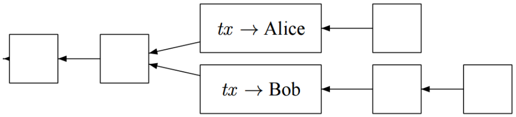
- $f(\text{previous block}, \text{current block}, \text{miner's id}, \text{nonce}) < c$
- A miner's chance of finding the next block is proportional to his computation power
- Most miners have very little power, compared to the whole network
- Miners' revenue has a high variance
- It's like winning a lottery that has positive expected value
- To reduce the variance, miners cooperate in pools
- A manager creates a pool, distributes hash inverting problems between miners, and divides the revenue among them
- Each miner receives a share proportional to the amount of work they did
- $f(\text{previous block}, \text{current block}, \text{pool manager's id}, \text{nonce}) < c'$ for some $c' > c$

Block Withholding Attack

- A miner can only turn partial solutions to the pool manager, but discard complete solutions
- Pools can and do attack each other

Double Spending is Still Possible

at least in theory



In order to double spend, Bob can:

- Create two transactions, one giving the money to Alice, the other one back to Bob
- Broadcast them at the same time from two nodes at different locations in the network, making sure that Alice sees the first transaction
- If Alice provides the service before seeing the second transaction, and the second transaction eventually gets into the consensus chain, the double spending attack is successful

In order to defend herself, Alice can wait for confirmations.

Fast Payments cannot be Confirmed

- A new block arrives every 10 minutes
- The usual practice is to wait for 6 confirmations (=1 hour)
- If Alice is selling a laptop, waiting for an hour before shipping is acceptable
- If Alice is a vending machine or a fast food restaurant, this is too much
- What else can Alice do?
  - She can put several nodes in different locations in the network to detect double spending
- How effective is this approach?
- It's basically a game between Alice and Bob!

Outline

Concurrent Games

A *concurrent stochastic game structure* $G = (S, A, \Gamma_1, \Gamma_2, \delta)$ has the following components:

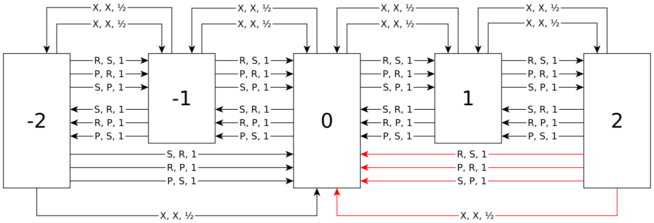
- A finite state space S and a finite set A of actions (or moves).
- Two move assignments $\Gamma_1, \Gamma_2 \colon S \to 2^A \setminus \emptyset$. For $i \in \{1, 2\}$, assignment Γ_i associates with each state $s \in S$ the non-empty set $\Gamma_i(s) \subseteq A$ of moves available to Player i at state s.
- A probabilistic transition function $\delta \colon S \times A \times A \to \mathcal{D}(S)$, which associates with every state $s \in S$ and moves $a_1 \in \Gamma_1(s)$ and $a_2 \in \Gamma_2(s)$, a probability distribution $\delta(s, a_1, a_2) \in \mathcal{D}(S)$ for the successor state.

Plays

At every state $s \in S$,

- Player 1 chooses a move $a_1 \in \Gamma_1(s)$,
- simultaneously and independently Player 2 chooses a move $a_2 \in \Gamma_2(s)$.
- The game then proceeds to the successor state t with probability $\delta(s, a_1, a_2)(t)$, for all $t \in S$.
- A *play* of G is an infinite sequence $\pi = \big((s_0, a_1^0, a_2^0), (s_1, a_1^1, a_2^1), (s_2, a_1^2, a_2^2) \dots\big)$ of states and action pairs such that for all $k \geq 0$ we have (i) $a_i^k \in \Gamma_i(s_k)$; and (ii) $s_{k+1} \in \mathrm{Supp}(\delta(s_k, a_1^k, a_2^k))$.
- Notation: We denote by Π the set of all plays.

Example



Strategies and Rewards

- We define a reward function $R : S \times A \times A \to \mathbb{R}$
- A strategy for Player i is a mapping $\sigma_i \colon (S \times A \times A)^* \times S \to \mathcal{D}(A)$
- An event in the game is a subset $A \subseteq \Pi$ of plays
- When a pair (σ_1, σ_2) of strategies are fixed, then the probabilities of measurable events are well-defined

Mean-payoff Objectives

- For a path $\pi = \big((s_0, a_1^0, a_2^0), (s_1, a_1^1, a_2^1), \dots\big)$, the average reward for T steps is $\mathrm{Avg}_T(\pi) = \frac{1}{T} \cdot \sum_{i=0}^{T-1} \mathrm{R}(s_i, a_1^i, a_2^i)$,
- The limit-inferior average is: $\mathrm{LimInfAvg}(\pi) = \liminf_{T\to\infty} \mathrm{Avg}_T(\pi)$
- The limit-superior average is: $\mathrm{LimSupAvg}(\pi) = \limsup_{T\to\infty} \mathrm{Avg}_T(\pi)$
- We consider a zero-sum game with mean-payoff objective
- The lower and upper game values at a state s are:

$$\underline{v}_s = \sup_{\sigma_1\in\Sigma_1} \inf_{\sigma_2\in\Sigma_2} \mathbb{E}_s^{\sigma_1,\sigma_2}[\mathrm{LimInfAvg}];$$

$$\overline{v}_s = \inf_{\sigma_2\in\Sigma_2} \sup_{\sigma_1\in\Sigma_1} \mathbb{E}_s^{\sigma_1,\sigma_2}[\mathrm{LimSupAvg}].$$

- *Determinacy:* $v_s := \underline{v}_s = \overline{v}_s$

Finding Values of Concurrent Games

- Determinacy was established in [Mertens and Neyman, 1981].
- Finite-memory strategies are not sufficient for optimality (e.g. Big Match [Gillete, 1957]).
- Given a state s, and a threshold λ, the problem of whether $v_s \geq \lambda$, can be decided in PSPACE [Chatterjee, Majumdar and Henzinger, 2008]
- All currently known algorithms use theory of reals and quantifier elimination and are not practical
- :(

Finding Values of Concurrent Games

- Determinacy was established in [Mertens and Neyman, 1981].
- Finite-memory strategies are not sufficient for optimality (e.g. Big Match [Gillete, 1957]).
- Given a state s, and a threshold λ, the problem of whether $v_s \geq \lambda$, can be decided in PSPACE [Chatterjee, Majumdar and Henzinger, 2008]
- All currently known algorithms use theory of reals and quantifier elimination and are not practical
- :(:(

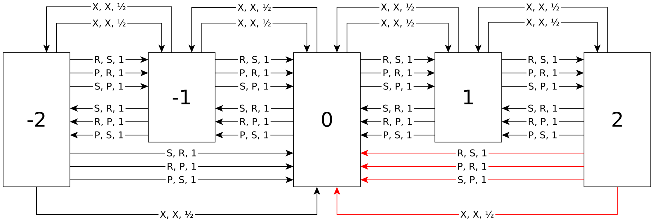
Finding Values of Concurrent Games

- Determinacy was established in [Mertens and Neyman, 1981].
- Finite-memory strategies are not sufficient for optimality (e.g. Big Match [Gillete, 1957]).
- Given a state s, and a threshold λ, the problem of whether $v_s \geq \lambda$, can be decided in PSPACE [Chatterjee, Majumdar and Henzinger, 2008]
- All currently known algorithms use theory of reals and quantifier elimination and are not practical
- :(:(:(
- How about looking into special classes of concurrent games?

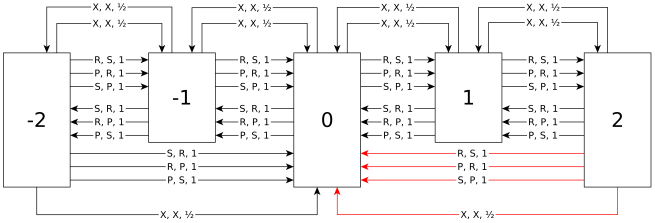
Ergodic Games

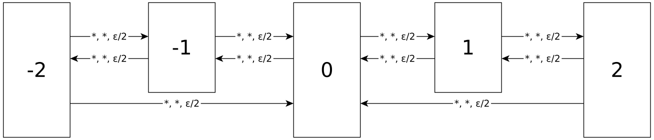
- A concurrent game G is *ergodic* if for all states $s, t \in S$, and all pairs of strategies (σ_1, σ_2), if we start at s, then t is visited infinitely often with probability 1 in the random walk $\pi_s^{\sigma_1,\sigma_2}$.
- Are real-world games ergodic?
- Can we solve ergodic games?

Back to Rock-Paper-Scissors



Back to Rock-Paper-Scissors





Solving Ergodic Games

We have the following results for Ergodic Games:

- Stationary optimal strategies exist [Hoffman and Karp, 1966]
- Values and probabilities of optimal strategies can be irrational [Chatterjee and Ibsen-Jensen, 2014], so the right question is to approximate them
- Strategy iteration converges [Hoffman and Karp, 1966]
- :) :) :)
- There was no practical implementation of the strategy iteration algorithm :(:(

Outline

Modeling Cryptocurrency Attacks as Ergodic Games

- Pool Attack:
  - There are two pools A and B that are attacking each other
  - At each turn, each of the two pools can decide how much of their mining power should be used to attack the other pool
  - States of the game correspond to the mining power of the pools
  - Miners are looking after their own interests and are not adversarial to either A or B
  - At each turn, some of the miners migrate between the pools A and B, or choose to mine for themselves. The migration is stochastic and depends on the revenue
  - The reward function R models the revenue of pool A
  - Stochastic migration makes the game ergodic

Modeling Cryptocurrency Attacks as Ergodic Games

- Zero-confirmation Double Spending:
  - Bob wants to buy a hamburger from Alice
  - Bob can choose to double spend the money or not
  - Alice can choose to wait for confirmation or not
  - Alice can choose to reset/change her connection to the network
  - The network evolves in small stochastic steps
  - The evolution of the network makes the game ergodic
  - The reward function R models Alice's revenue

Outline

Implementation

- We implemented strategy iteration for ergodic games and applied it to the cryptocurrency games
- There were two practical challenges:
  - Lack of Stopping Criteria
  - Numerical Precision

Results

#T	States	#SI	Time(s)
17050	100	4	69
56252	196	2	291
135252	289	2	389
236000	400	2	1059
331816	484	2	3880
508032	576	2	6273
720954	676	2	17014
966281	784	2	53103
1269450	900	2	100435

#T	States	#SI	Time(s)
19940	100	2	426
40040	200	2	800
60140	300	2	1141
80240	400	2	1586
100340	500	2	2069
120440	600	2	1253
140540	700	2	2999
160640	800	2	3496
180740	900	2	3917

Table: Experimental results for block-withholding pool attack (left) and zero-confirmation double-spending (right).