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

Krishnendu Chatterjee
IST Austria (Institute of Science and Technology Austria), Klosterneuburg, Austria
krishnendu.chatterjee@ist.ac.at

Amir Kafshdar Goharshady
IST Austria (Institute of Science and Technology Austria), Klosterneuburg, Austria
amir.goharshady@ist.ac.at

Rasmus Ibsen-Jensen
IST Austria (Institute of Science and Technology Austria), Klosterneuburg, Austria
ribsen@ist.ac.at

Yaron Velner
Hebrew University of Jerusalem, Jerusalem, Israel
yaron.welner@mail.huji.ac.il

Abstract

Crypto-currencies are digital assets designed to work as a medium of exchange, e.g., Bitcoin, but they are susceptible to attacks (dishonest behavior of participants). A framework for the analysis of attacks in crypto-currencies requires (a) modeling of game-theoretic aspects to analyze incentives for deviation from honest behavior; (b) concurrent interactions between participants; and (c) analysis of long-term monetary gains. Traditional game-theoretic approaches for the analysis of security protocols consider either qualitative temporal properties such as safety and termination, or the very special class of one-shot (stateless) games. However, to analyze general attacks on protocols for crypto-currencies, both stateful analysis and quantitative objectives are necessary. In this work our main contributions are as follows: (a) we show how a class of concurrent mean-payoff games, namely ergodic games, can model various attacks that arise naturally in crypto-currencies; (b) we present the first practical implementation of algorithms for ergodic games that scales to model realistic problems for crypto-currencies; and (c) we present experimental results showing that our framework can handle games with thousands of states and millions of transitions.

2012 ACM Subject Classification Software and its engineering → Formal software verification

Keywords and phrases Crypto-currency, Quantitative Verification, Mean-payoff Games

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2018.11

Related Version A full version of the paper is available at [10], `https://arxiv.org/abs/1806.03108`.

Acknowledgements The research was partially supported by Vienna Science and Technology Fund (WWTF) Project ICT15-003, Austrian Science Fund (FWF) NFN Grant No S11407-N23 (RiSE/SHiNE), ERC Starting Grant (279307: Graph Games), and an IBM PhD Fellowship.

1 Introduction

Economic effects of security violations. Traditionally, automated security analysis of protocols using game-theoretic frameworks focused on qualitative properties, such as safety or liveness [26, 16, 1], to ensure absolute security. In many cases absolute security is too expensive, and security violations are inevitable. In such scenarios rather than security, the economic implications of violations should be accounted for. In general, economic consequences of security violations are hard to measure. However, there is a new application area of crypto-currencies, in which the economic impact of an attack can be measured in terms of the number of coins that are lost. These currencies have considerable market value, in the order of hundreds of billions of dollars [18], thus developing a framework to formally analyze the security violations and their economic consequences for crypto-currencies is an interesting problem.

Crypto-currencies. There are many active crypto-currencies today, some with considerable market values. Currently, the main crypto-currency is Bitcoin with a value of over 150 billion dollars at the time of writing [18]. Virtually all of these currencies are free from outside governance and authority and are not controlled by any central bank. Instead, they work based on the decentralized *blockchain* protocol. This protocol, which was first developed for monetary transactions in Bitcoin [31], sets down the rules for creating new units of currency and valid transactions. However, it only defines the outcomes of actions taken by involved parties and cannot dictate the actions themselves. So, the whole ecosystem operates in a game-theoretic manner. The lack of an authority also leads to irreversibility of transactions, so if an amount of currency is transferred unintentionally or due to a bug, it cannot be reclaimed. This, together with the huge market values, makes it imperative to develop formal methods for quantifying the economic consequences before deploying the protocols.

Dishonest interaction. The fact that protocols define only the outcomes of actions and do not force the actions themselves, means that in some scenarios they might give one of the parties unfair or unintended advantage over others and an incentive to act dishonestly, i.e. to take an unintended action. Such behavior is called an attack. We succinctly describe some attacks.

- The most fundamental attack in every crypto-currency is *double-spending*, where one party could in some circumstances use the same coin twice in two different purchases. While this vulnerability is inherent in every blockchain protocol, people still use crypto-currencies as the probability (and the economic consequences) of such an attack can be bounded over time.
- Another line of attacks follow from dishonest behavior of the *blockchain miners* who are responsible for the underlying security of the blockchain protocol and are rewarded for their operations. It was shown that undesirable behavior, such as block withholding [19] or selfish mining [20], could increase the dishonest miner's reward, at the expense of other (honest) miners. We explain the block withholding attack in more detail in Section 5.1.

Research Questions. Analyzing attacks on crypto-currencies requires a formal framework to handle: (a) game-theoretic aspects and incentives for dishonest behavior; (b) simultaneous interaction of the participants; and (c) quantitative properties corresponding to long-term monetary gains and losses. These properties cannot be obtained from standard temporal or qualitative properties which have been the focus of previous game-theoretic frameworks [26, 16]. On the other hand, game-theoretic incentives are also analyzed in the security community (e.g., see [8]), but their methods are normally considering the very special case of one-shot (stateless) or short-term games. One-shot games cannot model the different states of the ecosystem or the history of actions taken.

Concurrent mean-payoff games. These games were introduced in the seminal work of Shapley [37], and later extended by Gillette [22]. A concurrent mean-payoff game is played by two players over a finite state space, where at each state both players simultaneously choose actions. The transition to the next state is determined by their joint actions, and each transition is assigned a reward. The goal of one player is to maximize the long-run average of the rewards, and the other player tries to minimize it. These games provide a very natural and general framework to study stateful games with simultaneous interactions and quantitative objectives. They lead to a very elegant and mathematically rich framework, and the theoretical complexity of such games has been studied for six decades [37, 22, 5, 24, 30, 14, 23]. However, the analysis of concurrent mean-payoff games is computationally intractable and no practical (such as strategy-iteration) algorithms exist to solve these games. Existing algorithmic approaches either require the theory of reals and quantifier elimination [14] or have doubly-exponential time complexity in the number of states [23].

Our contributions. Our main contributions are as follows:

1. **Modeling.** We propose to model long-term (infinite-horizon) economic aspects of security violations as concurrent mean-payoff games, between the attacker and the defender. The guaranteed payoff in the game corresponds to the maximal loss of the defender. In particular, for blockchain protocols, where the utility of every transition is naturally measurable, we show how to model various interesting scenarios as a sub-class of concurrent mean-payoff games, namely, *concurrent ergodic games*. In these games all states are visited infinitely often with probability 1.
2. **Practical implementation.** Second, while for concurrent ergodic games a theoretical algorithm (strategy-iteration algorithm) exists that does not use theory of reals and quantifier elimination, no previous implementation exists. Moreover, the implementation of the theoretical algorithm poses practical challenges: (a) the algorithm guarantees convergence only in the limit; and (b) the algorithm requires high numerical precision and the straightforward implementation of the algorithm does not converge in practice. We present (i) a simple stopping criterion for approximation, and (ii) resolve the numerical precision problem; and to our knowledge present the first practical implementation of a solver for concurrent ergodic games.
3. **Experimental results.** Finally, we present experimental results and show that the solver for ergodic games scales to thousands of states and nearly a million transitions to model realistic analysis problems from crypto-currencies. Note that in comparison, approaches for general concurrent mean-payoff games cannot handle even ten transitions (see the Remark in Section 3). Thus we present orders of magnitude of improvement.

2 Crypto-Currencies

Monetary system. A crypto-currency is a *monetary system* that allows secure transactions of currency units and dictates how new units are formed. Each transaction has a unique id and the following components: (i) a set of inputs; and (ii) a set of outputs and (iii) locking scripts. Each input has a pointer to an output of a previous transaction, and each output

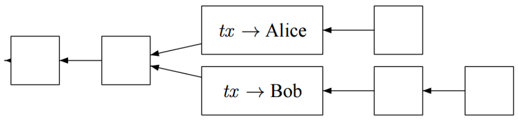

Figure 1 The longest chain dictates that the transaction tx belongs to Bob.

has an assigned monetary value. A locking script on an output defines a condition for using the funds stored in that output, e.g. the need for a digital signature. An input can only use funds of an output by passing its locking script.

Validity. A transaction is *valid* if these conditions hold: (a) the total value brought by the outputs is greater than or equal to the total value of the inputs; (b) the inputs have not been spent before; (c) the inputs satisfy locking scripts.

A transaction-based system is not secure if transactions are sent directly between users to transfer units. While validity conditions are enough to make sure that only valid recipients could redirect units they once truly held, there is nothing in the transactions themselves to limit the user from spending the same output twice (in two different transactions). For this purpose a public ledger of all valid transactions, called a *blockchain*, is maintained.

Blockchain. A ledger is a distributed database that maintains a growing *ordered* list of *valid* transactions. Its main novelty is that it enforces consensus among untrusted and possibly adversarial parties [31]. In Bitcoin (and most other major crypto-currencies) the public ledger is implemented as a series of *blocks* of transactions, each containing a reference to its previous block, and is hence called a blockchain. A consensus on the chain is obtained by a decentralized pseudonymous protocol. Any party tries to collect new transactions, form a block and add it to the chain (this process is called block mining). However, in order to do so, they must solve a challenging computational puzzle (which depends on the last block of the chain). The process of choosing the next block is as follows:

1. The first announced valid block that solves the puzzle is added to the chain.
2. If two valid blocks are found approximately at the same time (depending on network latency), then there is a temporary fork in the chain.

Every party is free to choose either fork, and try to extend it. Hence, the underlying structure of the blockchain is a tree. At any given time, the longest path in the tree, aka the *longest chain*, is the consensus blockchain (see Figure 1). Due to the random nature of the computational puzzle one branch will eventually become strictly longer than the other, and all parties will adopt it.

Mining process. The puzzle asks for a block consisting of valid transactions, hash of the previous block and an arbitrary integer *nonce*, whose hash is less than a target value. The random nature of the hash function dictates a simple strategy for mining: try random nonces until a solution is found. So the chance of a miner to find the next block is proportional to their computational power.

Incentives for mining. There are two incentives for miners: (i) Every transaction can donate to the miner who finds a new block that contains it, (ii) Each block creates a certain number of new coins which are then given to the miner.

Pool mining. To lower the variance of their revenue, miners often collaborate in *pools* [35, 8]. The pools have a manager who collects the rewards from valid blocks found by the members and allocates funds to them in proportion to the amount of work they did. Members prove their work by sending *partial solution* blocks, which are blocks with valid transactions but lower difficulty level, i.e., the hash of the block is not smaller than the network threshold, but it is lower than some threshold that was defined by the manager. As a result, pool members obtain lower variance in rewards, but have a small drop in expected revenue to cover the manager's fee. Members will get the same reward for a partial and full solution, but the member cannot claim the full block reward for themselves. More precisely, a block also dictates where the block reward goes to. Hence, even if a member broadcasts the new block, the reward will still go to the manager.

Proof of stake mining. An emerging criticism over the huge amount of energy that is wasted in the mining process led to development of *proof of stake protocols.* In proof of stake mining the miner is elected with probability that is proportional to their *stake* in the network (i.e., number of coin units he holds), rather than their computation power. Current proof of stake protocols assume a synchronous setting [32, 40, 28] where a miner is chosen in every time slot t_0. However, they differ in the way they reach consensus. We study a simplified version of [28].

1. At time t_0 a miner is randomly elected. She broadcasts the next block.
2. Until time $t_0 + t$ other miners who receive the block, verify it and if it were valid, sign it and broadcast the signature.
3. The block is added to the chain only if a majority of the network sign it.

To encourage honest behavior, the elected miner and signers get rewards when the suggested block is accepted.

3 Concurrent and Ergodic Games

Probability distributions. For a finite set A, a *probability distribution* on A is a function $\delta\colon A \to [0,1]$ such that $\sum_{a\in A} \delta(a) = 1$. We denote the set of probability distributions on A by $\mathcal{D}(A)$. Given a distribution $\delta \in \mathcal{D}(A)$, we denote by $\mathrm{Supp}(\delta) = \{x \in A \mid \delta(x) > 0\}$ the *support* of the distribution.

Concurrent game structures. 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.

We denote by n the number of states (i.e., $n = |S|$), and by m the maximal number of actions available for a player at a state (i.e., $m = \max_{s\in S} \max\{|\Gamma_1(s)|, |\Gamma_2(s)|\}$). The size of the transition relation of a game structure is defined as
$|\delta| = \sum_{s\in S} \sum_{a_1\in\Gamma_1(s)} \sum_{a_2\in\Gamma_2(s)} |\mathrm{Supp}(\delta(s, a_1, a_2))| \le n^2 \cdot m^2$.

Plays. At every state $s \in S$, Player 1 chooses a move $a_1 \in \Gamma_1(s)$, and 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 *path* or a *play* of G is an infinite

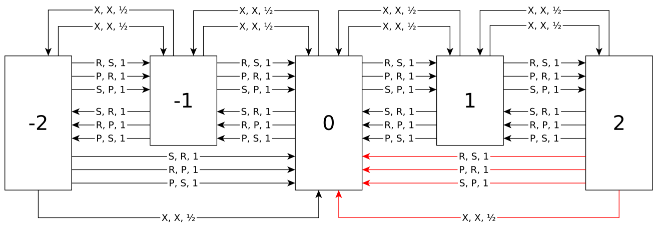

Figure 2 A repetitive rock-paper-scissors game.

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) $s_{k+1} \in \mathrm{Supp}(\delta(s_k, a_1^k, a_2^k))$. We denote by Π the set of all paths.

▶ **Example 1.** Consider a repetitive game of rock-paper-scissors, consisting of an infinite number of laps, in which each lap is made of a number of rounds as illustrated in Figure 2. When a lap begins, the two players play rock-paper-scissors repetitively until one of them wins 3 rounds more than her opponent, in which case she wins the current lap of the game and a new lap begins. In each round, the winner is determined by the usual rules of rock-paper-scissors, i.e. rock beats scissors, scissors beat paper and paper beats rock. In case of a tie, each player wins the round with probability $\frac{1}{2}$.

Here we have $S = \{-2, -1, 0, 1, 2\}$ and $\Gamma_1 = \Gamma_2 \equiv \{\mathrm{R}, \mathrm{P}, \mathrm{S}\}$. The game starts at state 0 and state s corresponds to the situation where Player 1 has won s rounds more than Player 2 in the ongoing lap. Edges in the figure correspond to possible transitions in the game. Each edge is labeled with three values a_1, a_2, p to denote that the game will transition from the state at the beginning of the edge to the state at its end with probability p if the two players decide on actions a_1 and a_2, respectively. For example, there is an edge from state 2 to state 0 labeled $\mathrm{R}, \mathrm{S}, 1$, which corresponds to $\delta(2, \mathrm{R}, \mathrm{S})(0) = 1$. In the figure, we use X, X in place of a_1, a_2 to denote that they are equal. Hence every *play* in this game corresponds to an infinite walk on the graph in Figure 2.

Strategies. A *strategy* is a recipe to extend prefixes of a play. Formally, a strategy for Player i is a mapping $\sigma_i \colon (S \times A \times A)^* \times S \to \mathcal{D}(A)$ that associates with every finite sequence $x \in (S \times A \times A)^*$ of state and action pairs, representing the past history of the game, and the current state s in S, a probability distribution $\sigma_i(x \cdot s)$ used to select the next move. The strategy σ_i can only prescribe moves that are available to Player i; that is, for all sequences $x \in (S \times A \times A)^*$ and states $s \in S$, we require $\mathrm{Supp}(\sigma_i(x \cdot s)) \subseteq \Gamma_i(s)$. We denote by Σ_i the set of all strategies for Player i. Once the starting state s and the strategies σ_1 and σ_2 for the two players have been chosen, then the probabilities of measurable events are uniquely defined [39]. For an event $\mathcal{A} \subseteq \Pi$, we denote by $\Pr_s^{\sigma_1,\sigma_2}(\mathcal{A})$ the probability that a path belongs to $\mathcal{A}$ when the game starts from s and the players use the strategies σ_1 and σ_2. We call a pair of strategies $(\sigma_1, \sigma_2) \in \Sigma_1 \times \Sigma_2$ a *strategy profile*.

Stationary (memoryless) and positional strategies. In general, strategies use randomization, and can use finite or even infinite memory to remember the history. Simpler strategies, that either do not use memory, or randomization, or both, are significant, as they are simple to implement and interpret. A strategy σ_i is *stationary* (or memoryless) if it is independent of the history but only depends on the current state, i.e., for all $x, x' \in (S \times A \times A)^*$ and all $s \in S$, we have $\sigma_i(x \cdot s) = \sigma_i(x' \cdot s)$, and thus can be expressed as a function $\sigma_i : S \to \mathcal{D}(A)$.

A strategy is *pure* if it does not use randomization, i.e., for any history there is a unique action a that is played with probability 1. A pure stationary strategy σ_i is called *positional*, and denoted as a function $\sigma_i : S \to A$.

Mean-payoff objectives. We consider maximizing *limit-average* (or mean-payoff) objectives for Player 1, and the objective of Player 2 is the opposite (i.e., the games are zero-sum). We consider concurrent games with a reward function $\mathrm{R} : S \times A \times A \to \mathbb{R}$ that assigns a reward value $\mathrm{R}(s, a_1, a_2)$ for all $s \in S$, $a_1 \in \Gamma_1(s)$, and $a_2 \in \Gamma_2(s)$. 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 for T steps is $\mathsf{Avg}_T(\pi) = \frac{1}{T} \cdot \sum_{i=0}^{T-1} \mathrm{R}(s_i, a_1^i, a_2^i)$, and the limit-inferior average (resp. limit-superior average) is defined as follows: $\mathsf{LimInfAvg}(\pi) = \liminf_{T\to\infty} \mathsf{Avg}_T(\pi)$ (resp. $\mathsf{LimSupAvg}(\pi) = \limsup_{T\to\infty} \mathsf{Avg}_T(\pi)$). We denote concurrent mean-payoff games as CMPGs.

▶ **Example 2.** Consider the game in Figure 2. In this game, Player 1 wins a lap whenever a red edge is crossed. Therefore, in order to capture the number of laps won by Player 1, rewards can be assigned as: $R(2, R, S) = R(2, P, R) = R(2, S, P) = 1$; $R(2, X, X) = \frac{1}{2}$ and 0 in all other cases.

Values and ϵ-optimal strategies. Given a CMPG G and a reward function R, the *lower value* $\underline{v}_s$ (resp. the *upper value* $\overline{v}_s$) at a state s is defined as follows:
$\underline{v}_s = \sup_{\sigma_1\in\Sigma_1} \inf_{\sigma_2\in\Sigma_2} \mathbb{E}_s^{\sigma_1,\sigma_2}[\mathsf{LimInfAvg}]$; $\overline{v}_s = \inf_{\sigma_2\in\Sigma_2} \sup_{\sigma_1\in\Sigma_1} \mathbb{E}_s^{\sigma_1,\sigma_2}[\mathsf{LimSupAvg}]$.

The *determinacy* result of [30] shows that the upper and lower values coincide and give the *value* of the game denoted as v_s. For $\epsilon \geq 0$, a strategy σ_1 for Player 1 is *ϵ-optimal* if we have $v_s - \epsilon \leq \inf_{\sigma_2\in\Sigma_2} \mathbb{E}_s^{\sigma_1,\sigma_2}[\mathsf{LimInfAvg}]$.

Ergodic Games. A CMPG G is *ergodic* if for all states $s, t \in S$, for all strategy profiles (σ_1, σ_2), if we start at s, then t is visited infinitely often with probability 1 in the random walk $\pi_s^{\sigma_1,\sigma_2}$. The game in Figure 2 is not ergodic. If Player 1 keeps playing rock and Player 2 scissors, then the states -1 and -2 are visited at most once. However, a more realistic version of this game is also ergodic.

▶ **Example 3.** Consider two players playing the repetitive game of rock-paper-scissors over a network, e.g. the Internet. The game is loaded on a central server that asks the players for their moves and provides them with rewards and information about changes in the state of the game. Given that the network is not perfect, there is always a small probability that one of the players is unable to announce his move in time to the server. In such cases, the player will lose the current round. Assume that this scenario happens with probability $\epsilon > 0$. Then all probabilities in Figure 2 have to be multiplied by $(1 - \epsilon)$ and new transitions, which are not under players' control and are a result of uncertainty in the network connection, should be added to the game. These new transitions are illustrated in Figure 3. Here a star can be replaced by any permissible action of the players. It is easy to check that this variant of the game is ergodic, given that starting from any state, there is a positive probability of visiting any other state within 3 steps using the new transitions only.

Results about general CMPGs. The main results for CMPGs are as follows:
1. The celebrated result of existence of values was established in [30].
2. For CMPGs, stationary or finite-memory strategies are not sufficient for optimality, and even in CMPGs with three states (the well-known Big Match game), very complex infinite-memory strategies are required for ϵ-optimality [5].

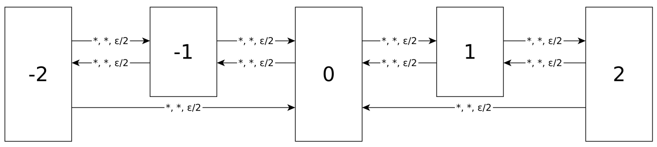

Figure 3 Transitions due to network connectivity issues in the repetitive RPS.

3. The value problem, that given a CMPG, a state s, and a threshold λ, asks whether the value at state s is at least λ, can be decided in PSPACE [14]; and also in $m^{2^{O(n)}}$ time, which is doubly exponential in the worst case, but polynomial-time in m, for n constant [23]. Both the above algorithms use the theory of reals and quantifier elimination for analysis.

▶ Remark (Inefficiency). The quantifier elimination approach for general CMPGs considers formulas in the theory of reals with alternation, where the variables represent the transitions [14]. With as few as ten transitions, quantifier elimination produces formulas with hundreds of variables over the existential theory of reals. In turn, the existential theory of reals has exponential-time complexity, is notoriously hard to solve, and its existing solvers cannot handle hundreds of variables. Hence, CMPGs with as few as ten transitions are not tractable.

Results about ergodic CMPGs. The main results for ergodic CMPGs are as follows:

1. Stationary optimal strategies exist[24], but positional strategies are not sufficient for optimality. For precise strategy complexity see [13].
2. Even in ergodic games, values and probabilities of optimal strategies can be irrational [13], and hence the relevant question is the approximation problem of values which is solvable in non-deterministic polynomial-time [13].
3. The most well-known algorithm for ergodic mean-payoff games is the Hoffman-Karp *strategy-iteration* algorithm [24]. See [10] for a more detailed treatment of this algorithm.

Note that since in ergodic games, every state is reached from every other state with probability 1, the value at all states is the same.

4 Modeling Framework

In this section we present an abstract framework to model economical consequences of attacks with mean-payoff games. In particular we show how broad classes of attacks can be modeled as ergodic games. In the next section we present concrete examples that arise from blockchain protocols.

4.1 Mean-payoff games modeling

We describe two aspects of mean-payoff games modeling.

1. *Game graph modeling.* Graph games are a standard model for reactive systems as well as protocols. The states and transitions of the graph represent states and transitions of the reactive system, and paths in the graphs represent traces of the system [33, 34]. Similarly, in modeling of protocols with different variables for the agents, the states of the game represent various scenarios of the protocols along with the valuation of the variables. The transitions represent a change of the scenario along with change in the valuation of the variables (for example see [16] for game graph modeling of protocols for digital-contract signing).

2. *Mean-payoff objective modeling.* In mean-payoff objectives, the costs (or rewards) of every transition can represent, for example, delays, execution times, cost of context switches, cost of concurrency, or monetary gains and losses. The mean-payoff objective represents the long-term average of the rewards or the costs. The mean-payoff objective has been used for synthesis of better reactive systems [7], synthesis of synchronization primitives for concurrent data-structures to minimize average context-switch costs [9], model resource-usage in container analysis and frequency of function calls [15], as well as analysis of energy-related objectives [3, 2, 21].

4.2 Crypto-currency Protocols as Mean-payoff Games

We describe how to apply the general framework of CMPGs to crypto-currencies:

- *General setting.* We propose to analyze protocols as a game between a defender and an attacker. The defender and the attacker have complete freedom to decide on their moves. The decisions of the other parties in the ecosystem can be modeled as stochastic choices that are not adversarial to either of the players.
- *Reward function.* The reward function will reflect the monetary gain or loss of the defender. The attacker gain is not modeled as we consider the worst-case scenario in which the attacker's objective is to minimize the defender's utility.
- *States.* States of the game can represent the information that is relevant for the analysis of the protocol, such as the abstract state of the blockchain.
- *Stochastic transitions.* Probabilities over the transitions can model true stochastic processes e.g., mining, or abstract complicated situations where the exact behavior cannot be directly computed (see Section 5.2) or in order to simulate the social behavior of a group (see Section 5.1).
- *Concurrent interactions.* Concurrent games are used when both players decide on their action simultaneously or when a single action models a behavior that continues over a time period and the players can only reason about their opponent's behavior after a while (Sections 5.1 and 5.2).
- *Result of the game.* In this work we want to reason on defender's security in a protocol wrt a malicious attacker who aims to decrease defender's gain at any cost. The result of the mean-payoff game will describe the inevitable expected loss that the defender will have in the presence of an attacker and defender's strategy describes the best way to defend himself against such an attacker.

4.3 Modeling with Ergodic Games

In this section we describe two classes of attacks, which can be naturally modeled with ergodic games. Our description here is high-level and informal, and concrete instances are considered in the next section. The attacks we describe are in a more general setting than crypto-currencies; however, for crypto-currencies the economic consequences are more natural to model.

First class of attacks. In the first class of attacks the setting consists of two companies and the revenues of the companies depend on the number of users each has. Thus states represent the number of users. Each company can decide to attack its competing company. Performing an attack entails some economic costs, however it could increase the number of users of the attacking company at the expense of the attacked one. For example, consider two competing social networks, Alice and Bob. Alice can decide to launch a distributed-denial-of-service

(DDOS) attack on Bob, and vice-versa. Such attacks entail a cost, but provide incentives for Bob users to switch to Alice. The rewards depend on the network revenues (i.e., number of users) and on the amount of funds the company decides to spend for the attack. The migration of users is a stochastic process that is biased towards the stronger network, but with smaller probability some users migrate to the other network. Thus the game is ergodic. This class represents pool attacks in the context of crypto-currencies (Sections 5.1 and 5.3).

Second class of attacks. Consider the scenario where the state of the game represents aspects of the dynamic network topology. The network evolves over the course of the time, and the actions of the participants also affect the network topology. However, the effect of the actions only makes local changes. The combination of the global changes and the local effects still ensure that different network states can be reached, and the game is ergodic. Attacks in such a scenario where the network topology determines the outcome of attack can be modeled as ergodic games. This class of attacks represent the zero-confirmation double-spending attack in the context of crypto-currencies (see Section 5.2).

5 Formal Modeling of Real Attacks

In this section we show how to model several real-world examples. These examples were described in the literature but were never analyzed as stateful games.

5.1 Block Withholding Pool Attack

Pools are susceptible to the classic block withholding attack [35], where a miner sends only partial solutions to the pool manager and discards full solutions. In this section we analyze block withholding attacks among two pools, pool A and pool B. We describe how pool A can attack pool B, and the converse direction is symmetric. To employ the pool block withholding attack, pool A registers at pool B as a regular miner. It receives tasks from pool B and transfers them to some of its own miners. Following the notions in [19], we call these infiltrating miners, and their mining power is called infiltration rate. When pool A's infiltrating miners deliver partial solutions, pool A's manager submits them to pool B's manager and proves the portion of work they did. When the infiltrating miners deliver a full solution, the attacking pool manager discards it.

At first, the total revenue of the victim pool does not change (as its effective mining rate was not changed), but the same sum is now divided among more miners. Thus, since the pool manager fees are nominal (fixed percentage of the total revenue [4]), in the short term, the manager of the victim pool will not lose. The attacker's mining power is reduced, since some of its miners are used for block withholding, but it earns additional revenue through its infiltration of the other pool. Finally, the total effective mining power in the system is reduced, causing the blockchain protocol to reduce the difficulty. Hence, in some scenarios, the attacker can gain, even in the short run, from performing the attack [19].

In the long run, if miners see a decrease in their profits (since they have to split the same revenue among more participants), it is likely that they consider to migrate to other pools. As a result, the victim pool's total revenue will decrease.

Our modeling. We aim to capture the long term consequences of pool attacks. We have two pools A and B, where B is the victim pool and A is the malicious pool who wishes to decrease B's profits. There is also a group of miners C who are honest and represent the rest of the network. In return, pool B can defend itself by attacking back. To simulate the long term effect, in every round pool members from A and B may migrate from one pool to another or to and from C. The migration is a stochastic process that favors the pool with maximum profitability for miners. We note that given sufficient amount of time (say a week), a pool manager can evaluate with very high probability the fraction of infiltrating miners in his pool. This can be done by looking at the ratio between full and partial solutions. Hence, in retrospect of a week, the pools are aware of each other's decisions, but within this week there is uncertainty. Therefore, we use concurrent games to analyze the worst case scenario for pool B.

▶ **Theorem 4.** *Consider a pair of pools A and B capable of attacking each other. Let C be the pool of remaining miners. If the miners in each pool migrate stochastically according to the attractiveness levels (as detailed below), then B can ensure a revenue of at least v on average per round, against any behavior of A, where v is the value of the concurrent ergodic game described below.*

5.1.1 Details of Modeling

We provide details of our modeling on some of the attacks to demonstrate how they can be thought of in terms of ergodic games. Details of all other attacks can be found in [10].

- *Game states.* We consider two pools, A and B and assume that any miner outside these two is mining independently for himself. Each state is defined by two values, i.e. the fractions of total computation power that belongs to A and B. We use a discretized version of this idea to model the game in a finite number of states and let $S = \{1, 2, \dots, n\}^2$ and define $\epsilon = \frac{1}{2n+1}$, where a state $(i_1, i_2) \in S$ corresponds to the case where pool A owns a fraction $\alpha_{i_1} = i_1\epsilon = \frac{i_1}{2n+1}$ of the total hash power and pool B controls a fraction $\beta_{i_2} = i_2\epsilon = \frac{i_2}{2n+1}$ of it. In this case the miners who work independently own a fraction $\gamma_{i_1,i_2} = 1 - \alpha_{i_1} - \beta_{i_2}$ of the total hash power.
- *Actions at each state.* Each pool can choose how much of its hash power it devotes to attacking the other pool. More formally, at each state $s = (i_1, i_2)$, pool A has i_1 choices of actions and $\Gamma_1(s) = \{a_1^0, a_1^1, a_1^2, \dots, a_1^{i_1-1}\}$ where a_1^j corresponds to attacking pool B with a fraction $j\epsilon$ of the total computing power of the network. Similarly $\Gamma_2(s) = \{a_2^0, a_2^1, a_2^2, \dots, a_2^{i_2-1}\}$.
- *Rewards.* We want the rewards to model the revenue (profit) of pool A, denoted by r_A, so we let $R(s, a_1^i, a_2^j) = r_A(s, a_1^i, a_2^j)$, for $a_1 \in \Gamma_1(s), a_2 \in \Gamma_2(s)$. We write r_A instead of $r_A(s, a_1^i, a_2^j)$ when there is no risk of confusion. We define r_B and r_C similarly and normalize the revenues: $r_A + r_B + r_C = 1$. To compute these values, we define "attractiveness". The attractiveness of a pool is its revenue divided by the total computing power of its miners. If pool A chooses the action a_1^i and pool B chooses the action a_2^j, then pool A is using a fraction $\alpha' = i\epsilon$ of the total network computing power to attack B and is receiving a corresponding fraction of B's revenue while not contributing to it. Therefore the attractiveness of pool B will be equal to: $attr_B = \frac{r_B}{\beta+\alpha'}$. Similarly we have $attr_A = \frac{r_A}{\alpha+\beta'}$, where $\beta' = j\epsilon$.
  Now consider the sources for pool A's revenue. It either comes from A's own mining process or from collecting shares of B's revenue, therefore:
  $$r_A = (\alpha - \alpha') + \alpha' \times attr_B,$$
  and similarly $r_B = (\beta - \beta') + \beta' \times attr_A$. The previous four equations provide us with a system of linear equations which we can solve to obtain the values of r_A, r_B, $attr_A$ and $attr_B$. Since a fraction $\alpha' + \beta'$ of total computation power is used on attacking other pools, we have: $attr_C = \frac{1}{1-\alpha'-\beta'}$.

- *Game transitions (δ).* Miners migrate between pools and a pool gains or loses mining power based on its attractiveness. If a pool is the most attractive option among the two, it gains ϵ new mining power with probability $\frac{2}{3}$, retains its current power with probability $\frac{1}{6}$ and loses ϵ power with probability $\frac{1}{6}$. On the other hand a pool that is not the most attractive option loses ϵ power with probability $\frac{2}{3}$, retains its current power with probability $\frac{1}{6}$ and attracts ϵ new mining power with probability $\frac{1}{6}$. These values were chosen for the purpose of demonstration of our algorithm and our implementation results. In practice, one can obtain realistic probabilities experimentally.
- *Ergodicity.* The game is ergodic because for each two states $s = (s_1, s_2)$ and $s' = (s'_1, s'_2)$ where $|s_1 - s'_1| \leq 1$ and $|s_2 - s'_2| \leq 1$, there is at least $\frac{1}{36}$ probability of going from s to s' no matter what choices the players make.

Proof of Theorem 4. Ergodicity was established in the final part above. The rest follows from the modeling and the determinacy result.

5.2 Zero-confirmation Double-spending

Nowadays, Bitcoin is increasingly used in “fast payments” such as online services, ATM withdrawals and vending machines [17], where the payment is followed by fast delivery of goods. While the blockchain consensus is appropriate for slow payments, it requires tens of minutes to confirm a transaction and is therefore inappropriate for fast payments. We consider a transaction confirmed when it is added to the blockchain and several blocks are added after it. This mechanism is essential for the detection of double-spending attacks in which an adversary attempts to use some of her coins for two or more payments. However, even in the absence of a confirmation, it is far from trivial to perform a double-spending attack. In a double spending attack, the attacker publishes two transactions that consume the same input. The attack is successful only if the victim node received one transaction and provided the goods before he became aware of the other, but eventually the latter was added to the blockchain. In an ideal world the attacker can increase his odds by broadcasting one transaction directly to the victim and the other at a far apart location, while on the other hand the victim can defend itself by deploying several nodes in the network in *strategic* locations. In the real world, however, the full topology of the network is never known to either of the parties. Nevertheless, based on history and network statistics one can estimate the odds of a successful attack given the current state of the network [6].

The victim has to decide on a policy for accepting zero-confirmation transactions. In particular he has to decide on the probability of whether to wait for a confirmation or not. If he waits for confirmation, then the payment is guaranteed, but customer satisfaction is damaged, and as a result the utility is smaller than the actual payment. If he does not wait for a confirmation, then the payment might be double spent. In the long term, the victim could decide to change the topology of the network. As it does not have full control over the topology, the outcome of the change is stochastic. Moreover, even when the victim does not initiate a change, the network topology is dynamic and keeps changing all the time. Hence, the odds of a successful attack are constantly changing in small stochastic steps.

Our modeling. We aim to analyze the worst case long run loss of the victim. In our model we abstract the network topology state and consider only the odds of successful double spending. We consider a scenario where the victim’s honest customers typically purchase goods worth 10 units per round. In every round, the victim decides on a policy for accepting fast payment, and the attacker, concurrently, unaware of the victim’s policy, has to decide the size of the attack. After every round, the victim decides if he wants to do a thorough change in the network topology. If he decides on a change, then the next state is chosen uniformly from all possible states (this represents the fact that neither players has full knowledge on the topology). If he decides to make no change, then the network state might still change, due to the dynamic nature of the network. In this case the next state is with high probability either the current state, or a state which is slightly better or slightly worse for the victim, but with low probability the state changes completely to an arbitrary state in the network (as sometimes small changes in the topology have big impact). The rewards stem from the outcome of each round in the following way: The payment is the sum of the honest customer purchases and the payment of the attacker (if it gets into the blockchain). The reward is the payment minus some penalty in case the victim has decided to wait for a confirmation. The fact that the network state is constantly changing makes our model ergodic.

▶ **Theorem 5** (Proof in [10])**.** *Consider a seller and an attacker in the zero-confirmation double spending problem. The seller can ensure profit of at least v on average per round, where v is the value of the corresponding CMPG.*

5.3 Proof of Stake Pool Attack

Proof of stake protocols let miners centralize their stakes in a pool. In such pools the withholding attack is not relevant as mining does not require physical resources. However, pool A might attack an opponent pool B by not signing or broadcasting its blocks. A successful attack would prevent the block from getting signed by a majority of the network and result in a loss of mining fees for B and can encourage miners to migrate from B. An unsuccessful attack decreases A's signing revenue.

Our modeling. We assume a setting similar to that of Section 5.1, where there are two opponent pools A and B, and the rest of the network consists of honest pools who sign every block that arrives on time. The states of the game are the stakes of each pool, namely α for pool A and β for pool B. In every round, with probability $1-(\alpha+\beta)$ neither of the pools is elected to mine a block, and no decisions are made. Otherwise, with probability $\frac{\alpha}{\alpha+\beta}$ pool A is elected and otherwise pool B is elected. When a pool is elected, the other pool decides whether to sign and broadcast the resulting block or not. In addition the network state and connectivity induce a distribution over the fraction of honest miners that receive the block. If the block is accepted, then its creator is rewarded with mining fees, and the other pool will get its signing fees only if it signed the block.

▶ **Theorem 6** (Proof in [10])**.** *Consider two pools A and B in a proof of stake mining system that can choose to attack each other by not signing blocks mined by the other pool. Consider that the rest of the network consists of independent miners who observe published blocks according to a predefined probability distribution and sign every valid block they observe. If the miners migrate according to the attractiveness levels (as described in Section 5.1), then B can ensure an average revenue of v against any behavior of A, where v is the value of the corresponding CMPG.*

6 Implementation and Experimental Results

Implementation. We have implemented the strategy-iteration algorithm for ergodic games (see [10] for pseudo-code and more details). The implementation is available at `http://ist.ac.at/~akafshda/concur2018`. To the best of our knowledge, this is the first implementation

Table 1 Experimental results for block-withholding pool attack (left), zero-confirmation double-spending (center) and proof of stake pool attack (right).

#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

#T	States	#SI	Time(s)
6076	99	18	471
20956	275	8	1338
31744	396	9	2520
44764	539	4	1073
77500	891	16	22125
119164	1331	27	32636
169756	1859	10	31597
262384	2816	12	89599

of this algorithm. The straightforward implementation of the strategy-iteration algorithm for ergodic games has two practical problems, which we describe below.

1. *No stopping criteria.* First, the strategy-iteration algorithm only guarantees convergence of values in the limit, and since values and probabilities in strategies can be irrational, convergence cannot be guaranteed in a finite number of steps. Hence we need a stopping criterion for approximation.
2. *Numerical precision issues.* Second, the stationary strategies in each iteration are obtained by solving LPs, which has numerical errors, and the probabilities sum to less than 1. If these errors remain, they cascade over iterations, and do not ensure convergence in practice for large examples. Hence we need to ensure numerical precision on top of the strategy-iteration algorithm.

Our solution for the above two problems are as follows:

1. *Stopping criteria.* We first observe that the value sequence which is obtained converges from below to the value of the game. In other words, the value sequence provide a lower bound to the lower value of the game. Hence we consider a symmetric version which is the strategy-iteration algorithm for player 2, and run each iteration of the two algorithms in sequence. The version for player 2 provides a lower bound on the lower value for player 2, and thus from that we can obtain an upper bound on the upper value of player 1. Since the upper and lower values coincide, we thus have both an upper and lower bound on the values, and once the difference is smaller than $\epsilon > 0$, then the algorithm has correctly approximated the value within ϵ and can stop and return the value and the strategy obtained as approximation.
2. *Numerical precision.* For numerical precision, instead of obtaining the results from the linear program, we obtain the set of *tight* and *slack* constraints, where the tight constraints represent the constraints where equality is obtained, and the other constraints are slack ones. From the tight constraints, which are equalities, we obtain the result using Gaussian elimination, which provides more precise values to the solution. We also tried other heuristics, such as adding the remaining probability to the greatest probability action, which led to similar results on convergence.

Experimental Results. Our experimental results are reported in Table 1. We show number of transitions in the game (#T), number of states in the game, the running time and number of strategy iterations (#SI). It is noteworthy that in all cases the number of iterations required is quite small. We also note that since the number of iterations is small, the crucial computational step is every iteration, where many LPs are solved. The outputs provided the following results (more details in [10]):

- For the block withholding pool attack game, the algorithm could guarantee a mean-payoff of 0.49 for the victim pool. In absence of an attacker the mean-payoff will be 1.
- For the zero-confirmation double-spending game, the algorithm verified that the seller is guaranteed to maintain at least half of her revenue, i.e., in presence of a malicious attacker, the value for the seller converges to 5 as the number of states increase, while it is 10 in absence of it.
- For the proof of stake pool attack game, by increasing the number of states, i.e., by refining the discretization, the guaranteed value (game value) decreases and tends to zero. In absence of an attacker, a pool A can achieve an expected payoff of $11s_A$ at a turn where s_A is the stake it holds.

7 Related Work

- *Pools attack.* The danger of a block withholding attack is as old as Bitcoin pools. The attack was described by Rosenfeld [35], as pools were becoming a dominant player in Bitcoin. While it was obvious that a pool is vulnerable to a malicious attacker, Eyal [19] showed that in some circumstances a pool can benefit by attacking another pool, and thus pool mining is vulnerable also in the presence of rational attackers. However, the analysis only considered the short term, i.e., the profit that the pool can get only in the short period after the attack. Laszka et al. [29] studied the long term impact of pools attack. In their framework miners are allowed to migrate from one pool to another. They analyzed the steady equilibrium in which the size of the pools become stable (although there is no guarantee that the game will converge to such a scenario). Our framework is the first to allow analysis of long term impacts without convergence assumptions.
- *Zero-confirmation double-spending.* Zero-confirmation double-spending was experimentally analyzed by Karame et al. [25] who gave numerical figures for the odds of successful double spending for different network states. However, their analysis did not consider that the victim may change his connectivity state. Our work is the first analysis of the long term impact of this attack.
- *Stateful analysis.* A stateful analysis of blockchain attacks was done by Sapirshtein et al. [36] and by Sompolinsky and Zohar [38]. In their analysis the different states of the blockchain were taken into account during the attack. The analysis was done using MDPs in which only the attacker decides on his actions and the victim follows a predefined protocol. A recent work [11] also considers abstraction-refinement for finite-horizon games based on smart contracts. However, it neither considers long-term behavior, nor mean-payoff objectives, nor can it model attacks such as double-spending and interactions between pools.
- *Quantitative verification with mean-payoff games.* The mean-payoff games problem has been studied extensively as a theoretical problem [33, 34]. It has also been studied in the context of verification and synthesis for performance related issues [7, 9, 15, 3, 2, 21]. However, all these works focus on turn-based games, and none of them consider concurrent games. To the best of our knowledge concurrent mean-payoff games have not been studied in the setting of security that we consider, where the quantitative objective is as crucial as safety critical issues. Practical implementation of algorithms for ergodic CMPGs do not exist in the literature.

8 Conclusion and Future Work

In this work we considered concurrent mean-payoff games, and in particular the subclass of ergodic games, to analyze attacks on crypto-currencies. There are several interesting directions to pursue: First, various notions of rationality are relevant to analyze games where the attacker is rational, rather than malicious, and aims to maximize his own utility instead of minimizing the defender's utility (e.g., secure-equilibria [12] or other related notions). Second, we consider two-player games, and the extension to multi-player games to model crypto-currency attacks is another interesting problem. Third, the modeling assumptions should be empirically validated and the parameters used to generate the games, e.g. the rates of migration, should be empirically obtained. Fourth, we consider the rest of the network to be neutral and stochastic. An interesting extension would be to consider a rational network, possibly consisting of coalitions of cooperating miners, as defined e.g. in [27].

References

1 M. Abadi and P. Rogaway. Reconciling two views of cryptography. In *Proceedings of the IFIP International Conference on Theoretical Computer Science*, pages 3–22. Springer, 2000.

2 C. Baier, C. Dubslaff, J. Klein, S. Klüppelholz, and S Wunderlich. Probabilistic model checking for energy-utility analysis. In *Horizons of the Mind. A Tribute to Prakash Panangaden - Essays Dedicated to Prakash Panangaden on the Occasion of His 60th Birthday*, pages 96–123, 2014.

3 C. Baier, S. Klüppelholz, H. de Meer, F. Niedermeier, and S. Wunderlich. Greener bits: Formal analysis of demand response. In *ATVA*, pages 323–339, 2016.

4 Bitcoin Wiki. Comparison of mining pools, 2017. URL: `http://en.bitcoin.it/Comparison_of_mining_pools`.

5 D. Blackwell and T. Ferguson. The big match. *The Annals of Mathematical Statistics*, 39(1):159–163, 1968.

6 blockcypher.com. Confidence factor, 2017. URL: `http://dev.blockcypher.com/#confidence-factor`.

7 R. Bloem, K. Chatterjee, T.A. Henzinger, and B. Jobstmann. Better quality in synthesis through quantitative objectives. In *CAV*, pages 140–156, 2009.

8 J. Bonneau, A. Miller, J. Clark, A. Narayanan, J.A. Kroll, and E.W. Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *IEEE Symposium on Security and Privacy*, pages 104–121. IEEE, 2015.

9 P. Cerný, K. Chatterjee, T.A. Henzinger, A. Radhakrishna, and R. Singh. Quantitative synthesis for concurrent programs. In *CAV*, pages 243–259, 2011.

10 K. Chatterjee, A.K. Goharshady, R. Ibsen-Jensen, and Y. Velner. Ergodic mean-payoff games for the analysis of attacks in crypto-currencies. *arXiv*, 2018. `arXiv:1806.03108`.

11 K. Chatterjee, A.K. Goharshady, and Y. Velner. Quantitative analysis of smart contracts. In *ESOP*, pages 739–767, 2018.

12 K. Chatterjee, T.A. Henzinger, and M. Jurdzinski. Games with secure equilibria. In *LICS*, pages 160–169, 2004.

13 K. Chatterjee and R. Ibsen-Jensen. The complexity of ergodic mean-payoff games. In *ICALP II*, pages 122–133, 2014.

14 K. Chatterjee, R. Majumdar, and T. A. Henzinger. Stochastic limit-average games are in EXPTIME. *Int. J. Game Theory*, 37(2):219–234, 2008.

15 K. Chatterjee, A. Pavlogiannis, and Y. Velner. Quantitative interprocedural analysis. In *POPL*, pages 539–551, 2015.

16 K. Chatterjee and V. Raman. Assume-guarantee synthesis for digital contract signing. *Formal Asp. Comput.*, 26(4):825–859, 2014.

17 CNN Money. Bitcoin's uncertain future as currency, 2011. URL: `http://money.cnn.com/video/technology/2011/07/18/t_bitcoin_currency.cnnmoney/`.

18 coinmarketcap.com. Crypto-currency market capitalizations, 2017. URL: `http://coinmarketcap.com/`.

19 I. Eyal. The miner's dilemma. In *IEEE Symposium on Security and Privacy*, pages 89–103. IEEE, 2015.

20 I. Eyal and E.G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, 2014.

21 V. Forejt, M. Z. Kwiatkowska, and D. Parker. Pareto curves for probabilistic model checking. In *ATVA*, pages 317–332, 2012.

22 D. Gillette. Stochastic games with zero stop probabilitites. In *CTG*, pages 179–188. Princeton University Press, 1957.

23 K. A. Hansen, M. Koucký, N. Lauritzen, P. B. Miltersen, and E. P. Tsigaridas. Exact algorithms for solving stochastic games: extended abstract. In *STOC*, pages 205–214, 2011.

24 A.J. Hoffman and R.M. Karp. On nonterminating stochastic games. *Management Sciences*, 12(5):359–370, 1966.

25 G. Karame, E. Androulaki, and S. Capkun. Two bitcoins at the price of one? double-spending attacks on fast payments in bitcoin. *IACR Cryptology ePrint Archive*, 2012:248, 2012.

26 S. Kremer and J.F. Raskin. A game-based verification of non-repudiation and fair exchange protocols. *Journal of Computer Security*, 2003.

27 Marta Z. Kwiatkowska, David Parker, and Aistis Simaitis. Strategic analysis of trust models for user-centric networks. In *SR*, pages 53–59, 2013.

28 J. Kwon. Tendermint: Consensus without mining, 2015. URL: `https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/`.

29 A. Laszka, B. Johnson, and J. Grossklags. When bitcoin mining pools run dry. In *International Conference on Financial Cryptography and Data Security*, pages 63–77. Springer, 2015.

30 J.F. Mertens and A. Neyman. Stochastic games. *IJGT*, 10:53–66, 1981.

31 S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

32 NxtCommunity. Nxt whitepaper, 2014. URL: `http://bravenewcoin.com/assets/Whitepapers/NxtWhitepaper-v122-rev4.pdf`.

33 A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190. ACM Press, 1989.

34 P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *IEEE Transactions on Control Theory*, 77:81–98, 1989.

35 Meni Rosenfeld. Analysis of bitcoin pooled mining reward systems. *arXiv preprint arXiv:1112.4980*, 2011.

36 A. Sapirshtein, Y. Sompolinsky, and A. Zohar. Optimal selfish mining strategies in bitcoin. *arXiv preprint arXiv:1507.06183*, 2015.

37 L.S. Shapley. Stochastic games. *PNAS*, 39:1095–1100, 1953.

38 Y. Sompolinsky and A. Zohar. Bitcoin's security model revisited. *CoRR*, abs/1605.09193, 2016.

39 M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state systems. In *FOCS*, pages 327–338. IEEE, 1985.

40 V. Zamfir. Introducing casper, the friendly ghost, 2015. URL: `https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/`.