

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

Krishnendu Chatterjee

IST Austria

krishnendu.chatterjee@ist.ac.at

Amir Kafshdar Goharshady

IST Austria

amir.goharshady@ist.ac.at

Rasmus Ibsen-Jensen

IST Austria

ribsen@ist.ac.at

Yaron Velner

Hebrew University of Jerusalem

aron.velner@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 → Software Verification and Validation → Formal Software Verification

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

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2018.

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 [32, 21, 2], 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 [23], thus developing a framework to formally analyze the security violations and their economic consequences for crypto-currencies is an interesting problem.



© Krishnendu Chatterjee, Amir K. Goharshady, Rasmus Ibsen-Jensen and Yaron Velner;
licensed under Creative Commons License CC-BY

Leibniz International Proceedings in Informatics
LIPIcs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

44 **Crypto-currencies.** There are many active crypto-currencies today, some with considerable market
 45 values. Currently, the main crypto-currency is Bitcoin with a value of over 150 billion dollars at the
 46 time of writing [23]. Virtually all of these currencies are free from outside governance and authority
 47 and are not controlled by any central bank. Instead, they work based on the decentralized *blockchain*
 48 protocol. This protocol, which was first developed for monetary transactions in Bitcoin [36], sets
 49 down the rules for creating new units of currency and valid transactions. However, it only defines
 50 the outcomes of actions taken by involved parties and cannot dictate the actions themselves. So,
 51 the whole ecosystem operates in a game-theoretic manner. The lack of an authority also leads to
 52 irreversibility of transactions, so if an amount of currency is transferred unintentionally or due to
 53 a bug, it cannot be reclaimed. This, together with the huge market values, makes it imperative to
 54 develop formal methods for quantifying the economic consequences before deploying the protocols.

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

- 60 ■ The most fundamental attack in every crypto-currency is *double-spending*, where one party could
 61 in some circumstances use the same coin twice in two different purchases. While this vulnerability
 62 is inherent in every blockchain protocol, people still use crypto-currencies as the probability (and
 63 the economic consequences) of such an attack can be bounded over time.
- 64 ■ Another line of attacks follow from dishonest behavior of the *blockchain miners* who are respons-
 65 ible for the underlying security of the blockchain protocol and are rewarded for their operations.
 66 It was shown that undesirable behavior, such as block withholding [24] or selfish mining [25],
 67 could increase the dishonest miner's reward, at the expense of other (honest) miners. We explain
 68 the block withholding attack in more detail in Section 5.1.

69 **Research Questions.** Analyzing attacks on crypto-currencies requires a formal framework to handle:
 70 (a) game-theoretic aspects and incentives for dishonest behavior; (b) simultaneous interaction of the
 71 participants; and (c) quantitative properties corresponding to long-term monetary gains and losses.
 72 These properties cannot be obtained from standard temporal or qualitative properties which have
 73 been the focus of previous game-theoretic frameworks [32, 21]. On the other hand, game-theoretic
 74 incentives are also analyzed in the security community (e.g., see [13]), but their methods are normally
 75 considering the very special case of one-shot (stateless) or short-term games. One-shot games cannot
 76 model the different states of the ecosystem or the history of actions taken by participants.

77 **Concurrent mean-payoff games.** These games were introduced in the seminal work of Shapley [44],
 78 and later extended by Gillette [28]. A concurrent mean-payoff game is played by two players over
 79 a finite state space, where at each state both players simultaneously choose actions. The transition
 80 to the next state is determined by their joint actions, and each transition is assigned a reward.
 81 The goal of one player is to maximize the long-run average of the rewards, and the other player
 82 tries to minimize it. These games provide a very natural and general framework to study stateful
 83 games with simultaneous interactions and quantitative objectives. They lead to a very elegant and
 84 mathematically rich framework, and the theoretical complexity of such games has been studied for
 85 six decades [44, 28, 9, 30, 35, 19, 29]. However, the analysis of concurrent mean-payoff games is
 86 computationally intractable and no practical (such as strategy-iteration) algorithms exist to solve
 87 these games. Existing algorithmic approaches either require the theory of reals and quantifier
 88 elimination [19] or have doubly-exponential time complexity in the number of states [29], and cannot
 89 handle beyond toy examples of ten transitions.

90 **Our contributions.** Our main contributions are as follows:

- 91 1. **Modeling.** We propose to model long-term (infinite-horizon) economic aspects of security viola-
 92 tions as concurrent mean-payoff games, between the attacker and the defender. The guaranteed
 93 payoff in the game corresponds to the maximal loss of the defender. In particular, for blockchain

94 protocols, where the utility of every transition is naturally measurable, we show how to model
 95 various interesting scenarios as a sub-class of concurrent mean-payoff games, namely, *concurrent*
 96 *ergodic games*. In these games all states are visited infinitely often with probability 1.

97 **2. Practical implementation.** Second, while for concurrent ergodic games a theoretical algorithm
 98 (strategy-iteration algorithm) exists that does not use theory of reals and quantifier elimination,
 99 no previous implementation exists. Moreover, the implementation of the theoretical algorithm
 100 poses practical challenges: (a) the algorithm guarantees convergence only in the limit; and
 101 (b) the algorithm requires high numerical precision and the straightforward implementation
 102 of the algorithm does not converge in practice. We present (i) a simple stopping criterion for
 103 approximation, and (ii) resolve the numerical precision problem; and to our knowledge present
 104 the first practical implementation of a solver for concurrent ergodic games.

105 **3. Experimental results.** Finally, we present experimental results and show that the solver for
 106 ergodic games scales to thousands of states and nearly a million transitions to model realistic
 107 analysis problems from crypto-currencies. Note that in comparison, approaches for general
 108 concurrent mean-payoff games cannot handle even ten transitions (see the Remark in Section 3).
 109 Thus we present orders of magnitude of improvement.

110 2 Crypto-Currencies

111 **Monetary system.** A crypto-currency is a *monetary system* that allows secure transactions of currency
 112 units and dictates how new units are formed. Each transaction has a unique id and the following
 113 components: (i) a set of inputs; and (ii) a set of outputs and (iii) locking scripts. Each input has a
 114 pointer to an output of a previous transaction, and each output has an assigned monetary value. A
 115 locking script on an output defines a condition for using the funds stored in that output, e.g. the need
 116 for a digital signature. An input can use funds of the output it points to only if it can satisfy this
 117 condition.

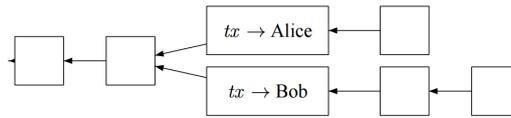
118 **Validity.** A transaction is *valid* if these conditions hold: (a) the total value brought by the inputs is
 119 greater than or equal to the total value of the outputs; (b) the inputs have not been spent before; (c)
 120 the inputs satisfy locking scripts.

121 Note that the list of transactions is the only state of the system and higher level concepts like
 122 account balance and users are computed directly from it. A transaction-based system is not secure if
 123 transactions are sent directly between users to transfer units. While validity conditions are enough
 124 to make sure that only valid recipients could redirect units they once truly held, there is nothing in
 125 the transactions themselves to limit the user from spending the same output twice (in two different
 126 transactions). For this purpose a public ledger of all valid transactions, called a *blockchain*, is
 127 maintained.

128 **Blockchain.** A ledger is a distributed database that maintains a growing *ordered* list of *valid*
 129 transactions. Its main novelty is that it enforces consensus among untrusted and possibly adversarial
 130 parties [36]. In Bitcoin (and most other major crypto-currencies) the public ledger is implemented as
 131 a series of *blocks* of transactions, each containing a reference to its previous block, and is hence called
 132 a blockchain. A consensus on the chain is obtained by a decentralized pseudonymous protocol. Any
 133 party tries to collect new transactions, form a block and add it to the chain (this process is called block
 134 mining). However, in order to do so, they must solve a challenging computational puzzle (which
 135 depends on the last block of the chain). The process of choosing the next block is as follows:

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

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



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

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

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

150 **Pool mining.** To lower the variance of their revenue, miners often collaborate in *pools* [40, 13]. The
 151 pools have a manager who collects the rewards from valid blocks found by the members and allocates
 152 funds to them in proportion to the amount of work they did. Members prove their work by sending
 153 *partial solution* blocks, which are blocks with valid transactions but lower difficulty level, i.e., the
 154 hash of the block is not smaller than the network threshold, but it is lower than some threshold that
 155 was defined by the manager. As a result, pool members obtain lower variance in rewards, but have a
 156 small drop in expected revenue to cover the manager’s fee. Members will get the same reward for a
 157 partial and full solution, but the member cannot claim the full block reward for themselves. More
 158 precisely, a block also dictates where the block reward goes to. Hence, even if a member broadcasts
 159 the new block, the reward will still go to the manager.

160 **Overview.** A crypto-currency is a network with nodes. Some of the nodes are also miners. A node
 161 has a local copy of the blockchain and local *transaction pool*, which holds valid pending transactions
 162 that are still not in the blockchain. When a user performs a transaction his associated nodes broadcast
 163 the transaction to the network. When a node receives a new transaction it checks whether it is valid
 164 wrt its blockchain and transaction pool. When a node receives a new block, it verifies that it is valid
 165 wrt consensus chain. If it is valid it adds it to the chain and updates his transaction pool accordingly.
 166 Whenever a new valid transaction or block is received, the node broadcasts it to all of its neighbors.

167 **Proof of stake mining.** An emerging criticism over the huge amount of energy that is wasted in the
 168 mining process led to development of *proof of stake protocols*. In proof of stake mining the miner is
 169 elected with probability that is proportional to their *stake* in the network (i.e., number of coin units he
 170 holds), rather than their computation power. Current proof of stake protocols assume a synchronous
 171 setting [37, 47, 33] where a miner is chosen in every time slot t_0 . However, they differ in the way
 172 they reach consensus. We study a simplified version of [33].

- 173 1. At time t_0 a miner is randomly elected. She broadcasts the next block.
 - 174 2. Until time $t_0 + t$ other miners who receive the block, verify it and if it were valid, sign it and
 175 broadcast the signature.
 - 176 3. The block is added to the chain only if a majority of the network sign it.
- 177 To encourage honest behavior, the elected miner and signers get rewards when the suggested block is
 178 accepted.

179 3 Concurrent and Ergodic Games

180 We first present the basic definitions and results related to concurrent games.

181 **Probability distributions.** For a finite set A , a *probability distribution* on A is a function $\delta: A \rightarrow$
 182 $[0, 1]$ such that $\sum_{a \in A} \delta(a) = 1$. We denote the set of probability distributions on A by $\mathcal{D}(A)$. Given a
 183 distribution $\delta \in \mathcal{D}(A)$, we denote by $\text{Supp}(\delta) = \{x \in A \mid \delta(x) > 0\}$ the *support* of the distribution.

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

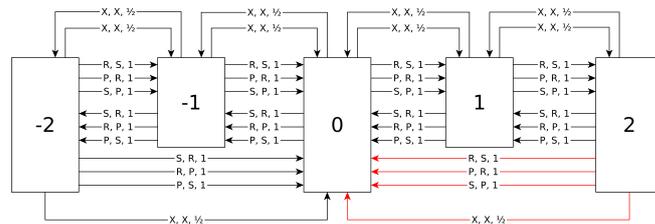
- 186 ■ A finite state space S and a finite set A of actions (or moves).
- 187 ■ Two move assignments $\Gamma_1, \Gamma_2: S \rightarrow 2^A \setminus \emptyset$. For $i \in \{1, 2\}$, assignment Γ_i associates with each
 188 state $s \in S$ the non-empty set $\Gamma_i(s) \subseteq A$ of moves available to Player i at state s .
- 189 ■ A probabilistic transition function $\delta: S \times A \times A \rightarrow \mathcal{D}(S)$, which associates with every state
 190 $s \in S$ 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
 191 the successor state.

192 We denote by n the number of states (i.e., $n = |S|$), and by m the maximal number of actions available
 193 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
 194 of a game structure is defined as $|\delta| = \sum_{s \in S} \sum_{a_1 \in \Gamma_1(s)} \sum_{a_2 \in \Gamma_2(s)} |\text{Supp}(\delta(s, a_1, a_2))| \leq n^2 \cdot m^2$.

195 **Plays.** At every state $s \in S$, Player 1 chooses a move $a_1 \in \Gamma_1(s)$, and simultaneously and
 196 independently Player 2 chooses a move $a_2 \in \Gamma_2(s)$. The game then proceeds to the successor
 197 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 sequence
 198 $\pi = ((s_0, a_1^0, a_2^0), (s_1, a_1^1, a_2^1), (s_2, a_1^2, a_2^2) \dots)$ of states and action pairs such that for all $k \geq 0$ we
 199 have (i) $a_i^k \in \Gamma_i(s_k)$; and (ii) $s_{k+1} \in \text{Supp}(\delta(s_k, a_1^k, a_2^k))$. We denote by Π the set of all paths.

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

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



■ **Figure 2** A repetitive rock-paper-scissors game

214 **Strategies.** A *strategy* is a recipe to extend prefixes of a play. Formally, a strategy for Player i is a
 215 mapping $\sigma_i: (S \times A \times A)^* \times S \rightarrow \mathcal{D}(A)$ that associates with every finite sequence $x \in (S \times A \times A)^*$
 216 of state and action pairs, representing the past history of the game, and the current state s in S , a
 217 probability distribution $\sigma_i(x \cdot s)$ used to select the next move. The strategy σ_i can only prescribe
 218 moves that are available to Player i ; that is, for all sequences $x \in (S \times A \times A)^*$ and states $s \in S$,
 219 we require $\text{Supp}(\sigma_i(x \cdot s)) \subseteq \Gamma_i(s)$. We denote by Σ_i the set of all strategies for Player i . Once the
 220 starting state s and the strategies σ_1 and σ_2 for the two players have been chosen, then the probabilities

221 of measurable events are uniquely defined [46]. For an event $\mathcal{A} \subseteq \Pi$, we denote by $\Pr_s^{\sigma_1, \sigma_2}(\mathcal{A})$ the
 222 probability that a path belongs to \mathcal{A} when the game starts from s and the players use the strategies σ_1
 223 and σ_2 ; and $\mathbb{E}_s^{\sigma_1, \sigma_2}[\cdot]$ is the expectation measure. We call a pair of strategies $(\sigma_1, \sigma_2) \in \Sigma_1 \times \Sigma_2$ a
 224 *strategy profile*.

225 **Stationary (memoryless) and positional strategies.** In general, strategies use randomization, and
 226 can use finite or even infinite memory to remember the history. Simpler strategies, that either do not
 227 use memory, or randomization, or both, are significant, as they are simple to implement and interpret.
 228 A strategy σ_i is *stationary* (or memoryless) if it is independent of the history but only depends on the
 229 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
 230 can be expressed as a function $\sigma_i : S \rightarrow \mathcal{D}(A)$. A strategy is *pure* if it does not use randomization,
 231 i.e., for any history there is always some unique action a that is played with probability 1. A pure
 232 stationary strategy σ_i is called *positional*, and represented as a function $\sigma_i : S \rightarrow A$.

233 **Mean-payoff objectives.** We consider maximizing *limit-average* (or mean-payoff) objectives for
 234 Player 1, and the objective of Player 2 is the opposite (i.e., the games are zero-sum). We consider
 235 concurrent games with a reward function $R : S \times A \times A \rightarrow \mathbb{R}$ that assigns a reward value $R(s, a_1, a_2)$
 236 for all $s \in S$, $a_1 \in \Gamma_1(s)$, and $a_2 \in \Gamma_2(s)$. For a path $\pi = ((s_0, a_1^0, a_2^0), (s_1, a_1^1, a_2^1), \dots)$,
 237 the average for T steps is $\text{Avg}_T(\pi) = \frac{1}{T} \cdot \sum_{i=0}^{T-1} R(s_i, a_1^i, a_2^i)$, and the limit-inferior average
 238 (resp. limit-superior average) is defined as follows: $\text{LimInfAvg}(\pi) = \liminf_{T \rightarrow \infty} \text{Avg}_T(\pi)$ (resp.
 239 $\text{LimSupAvg}(\pi) = \limsup_{T \rightarrow \infty} \text{Avg}_T(\pi)$). For brevity we denote concurrent games with mean-
 240 payoff objectives as CMPGs (concurrent mean-payoff games).

241 ► **Example 2.** Consider the game in Figure 2. In this game, Player 1 wins a lap whenever a red
 242 edge is crossed. Therefore, in order to capture the number of laps won by Player 1, rewards can be
 243 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.

244 **Values and ϵ -optimal strategies.** Given a CMPG G and a reward function R , the *lower value* \underline{v}_s
 245 (resp. the *upper value* \bar{v}_s) at a state s is defined as follows:

$$246 \quad \underline{v}_s = \sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} \mathbb{E}_s^{\sigma_1, \sigma_2}[\text{LimInfAvg}]; \quad \bar{v}_s = \inf_{\sigma_2 \in \Sigma_2} \sup_{\sigma_1 \in \Sigma_1} \mathbb{E}_s^{\sigma_1, \sigma_2}[\text{LimSupAvg}].$$

247 The *determinacy* result of [35] shows that the upper and lower values coincide and give the *value*
 248 of the game denoted as v_s . For $\epsilon \geq 0$, a strategy σ_1 for Player 1 is ϵ -*optimal* if we have $v_s - \epsilon \leq$
 249 $\inf_{\sigma_2 \in \Sigma_2} \mathbb{E}_s^{\sigma_1, \sigma_2}[\text{LimInfAvg}]$.

250 **Ergodic Games.** A CMPG G is *ergodic* if for all states $s, t \in S$, for all strategy profiles (σ_1, σ_2) , if
 251 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
 252 in Figure 2 is not ergodic. If Player 1 keeps playing rock and Player 2 scissors, then the states -1 and
 253 -2 are visited at most once each. We now present a more realistic version of the same game that is
 254 also ergodic.

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

266 **Results about general CMPGs.** The main results for CMPGs are as follows:

267 **1.** The celebrated result of existence of values was established in [35].

268 2. For CMPGs, stationary or finite-memory strategies are not sufficient for optimality, and even
 269 in CMPGs with three states (the well-known Big Match game), very complex infinite-memory
 270 strategies are required for ϵ -optimality [9].

271 3. The value problem, that given a CMPG, a state s , and a threshold λ , asks whether the value at
 272 state s is at least λ , can be decided in PSPACE [19]; and also in $m^{2^{O(n)}}$ time, which is doubly
 273 exponential in the worst case, but polynomial-time in m , for n constant [29]. Both the above
 274 algorithms use the theory of reals and quantifier elimination for analysis.

275 ► Remark (Inefficiency). The quantifier elimination approach for general CMPGs considers
 276 formulas in the theory of reals with alternation, where the variables represent the transitions [19].
 277 With as few as ten transitions, quantifier elimination produces formulas with hundreds of variables
 278 over the existential theory of reals. In turn, the existential theory of reals has exponential-time
 279 complexity, is notoriously hard to solve, and its existing solvers cannot handle hundreds of variables.
 280 Hence, CMPGs with as few as ten transitions are not tractable.

281 **Results about ergodic CMPGs.** The main results for ergodic CMPGs, besides the general results
 282 for CMPGs, are as follows:

283 1. Stationary optimal strategies exist [30], but positional strategies are not sufficient for optimality.
 284 For precise strategy complexity see [18].

285 2. Even in ergodic games, values and probabilities of optimal strategies can be irrational [18],
 286 and hence the relevant question is the approximation problem of values which is solvable in
 287 non-deterministic polynomial-time [18].

288 3. The most well-known algorithm for ergodic mean-payoff games is the Hoffman-Karp *strategy-*
 289 *iteration* algorithm [30], which is described in detail in Appendix A.

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

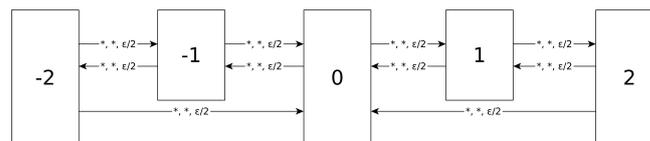
292 4 Modeling Framework

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

297 4.1 Mean-payoff games modeling

298 We describe two aspects of mean-payoff games modeling.

299 1. *Game graph modeling.* Graph games are a standard model for reactive systems as well as
 300 protocols. The states and transitions of the graph represent states and transitions of the reactive
 301 system, and paths in the graphs represent traces of the system [38, 39]. Similarly, in modeling of
 302 protocols with different variables for the agents, the states of the game represent various scenarios
 303 of the protocols along with the valuation of the variables. The transitions represent a change of
 304 the scenario along with change in the valuation of the variables (for example see [21] for game
 305 graph modeling of protocols for digital-contract signing).



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

306 **2. Mean-payoff objective modeling.** In mean-payoff objectives, the costs (or rewards) of every
 307 transition can represent, for example, delays, execution times, cost of context switches, cost of
 308 concurrency, or monetary gains and losses. The mean-payoff objective represents the long-term
 309 average of the rewards or the costs. The mean-payoff objective has been used for synthesis of
 310 better reactive systems [12], synthesis of synchronization primitives for concurrent data-structures
 311 to minimize average context-switch costs [15], model resource-usage in container analysis and
 312 frequency of function calls [20], as well as analysis of energy-related objectives [7, 6, 26].

313 4.2 Crypto-currency Protocols as Mean-payoff Games

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

315 **General setting.** We propose to analyze protocols as a game between a defender and an attacker. The
 316 defender and the attacker have complete freedom to decide on their moves. The decisions of the other
 317 parties in the ecosystem can be modeled as stochastic choices that are not adversarial to either of the
 318 players.

319 **Reward function.** The reward function will reflect the monetary gain or loss of the defender. The
 320 attacker gain is not modeled as we consider the worst-case scenario in which the attacker's objective
 321 is to minimize the defender's utility.

322 **States.** States of the game can represent the information that is relevant for the analysis of the
 323 protocol, such as the abstract state of the blockchain.

324 **Stochastic transitions.** Probabilities over the transitions can model true stochastic processes e.g.,
 325 mining, or abstract complicated situations where the exact behavior cannot be directly computed (see
 326 Section 5.2) or in order to simulate the social behavior of a group (see Section 5.1).

327 **Concurrent interactions.** Concurrent games are used when both players need to decide on their
 328 action simultaneously or when a single action models a behavior that continues over a time period
 329 and the players can only reason about their opponent's behavior after some while (see Sections 5.1
 330 and 5.2).

331 **Result of the game.** In this work we want to reason on defender's security in a protocol wrt a
 332 malicious attacker who aims to decrease defender's gain at any cost. The result of the mean-payoff
 333 game will describe the inevitable expected loss that the defender will have in the presence of an
 334 attacker and defender's strategy describes the best way to defend himself against such an attacker.

335 4.3 Modeling with Ergodic Games

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

340 *First class of attacks.* In the first class of attacks the setting consists of two companies and the
 341 revenues of the companies depend on the number of users each has. Thus states represent the number
 342 of users. Each company can decide to attack its competing company. Performing an attack entails
 343 some economic costs, however it could increase the number of users of the attacking company at the
 344 expense of the attacked one. For example, consider two competing social networks, Alice and Bob.
 345 Alice can decide to launch a distributed-denial-of-service (DDOS) attack on Bob, and vice-versa.
 346 Such attacks entail a cost, but provide incentives for Bob users to switch to Alice. The rewards depend
 347 on the network revenues (i.e., number of users) and on the amount of funds the company decides to
 348 spend for the attack. The migration of users is a stochastic process that is biased towards the stronger
 349 network, but with smaller probability some users migrate to the other network. Thus the game is
 350 ergodic. This class represents pool attacks in the context of crypto-currencies (Sections 5.1 and 5.3).

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

358 **5 Formal Modeling of Real Attacks**

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

361 **5.1 Block Withholding Pool Attack**

362 Pools are susceptible to the classic block withholding attack [40], where a miner sends only partial
363 solutions to the pool manager and discards full solutions. In this section we analyze block withholding
364 attacks among two pools, pool A and pool B . We describe how pool A can attack pool B , and the
365 converse direction is symmetric. To employ the pool block withholding attack, pool A registers at
366 pool B as a regular miner. It receives tasks from pool B and transfers them to some of its own miners.
367 Following the notions in [24], we call these infiltrating miners, and their mining power is called
368 infiltration rate. When pool A 's infiltrating miners deliver partial solutions, pool A 's manager submits
369 them to pool B 's manager and proves the portion of work they did. When the infiltrating miners
370 deliver a full solution, the attacking pool manager discards it.

371 At first, the total revenue of the victim pool does not change (as its effective mining rate was not
372 changed), but the same sum is now divided among more miners. Thus, since the pool manager fees
373 are nominal (fixed percentage of the total revenue [8]), in the short term, the manager of the victim
374 pool will not lose. The attacker's mining power is reduced, since some of its miners are used for
375 block withholding, but it earns additional revenue through its infiltration of the other pool. Finally,
376 the total effective mining power in the system is reduced, causing the blockchain protocol to reduce
377 the difficulty. Hence, in some scenarios, the attacker can gain, even in the short run, from performing
378 the attack [24].

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

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

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

397 **5.1.1 Details of Modeling**

398 We provide details of our modeling to demonstrate how such attacks can be thought of in terms of
 399 ergodic games. Due to page limitation and similarity, such details in other cases are relegated to
 400 Appendix C.

401 ■ *Game states.* We consider two pools, A and B and assume that any miner outside these two is
 402 mining independently for himself. Each state is defined by two values, i.e. the fractions of total
 403 computation power that belongs to A and B . We use a discretized version of this idea to model
 404 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
 405 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
 406 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
 407 who work independently own a fraction $\gamma_{i_1, i_2} = 1 - \alpha_{i_1} - \beta_{i_2}$ of the total hash power.

408 ■ *Actions at each state.* Each pool can choose how much of its hash power it devotes to attacking
 409 the other pool. More formally, at each state $s = (i_1, i_2)$, pool A has i_1 choices of actions and
 410 $\Gamma_1(s) = \{a_1^0, a_1^1, a_1^2, \dots, a_1^{i_1-1}\}$ where a_1^j corresponds to attacking pool B with a fraction $j\epsilon$
 411 of the total computing power of the network. Similarly $\Gamma_2(s) = \{a_2^0, a_2^1, a_2^2, \dots, a_2^{i_2-1}\}$.

412 ■ *Rewards.* We want the rewards to model the revenue (profit) of pool A , denoted by r_A , so we let
 413 $R(s, a_1^i, a_2^j) = r_A(s, a_1^i, a_2^j)$, for $a_1 \in \Gamma_1(s), a_2 \in \Gamma_2(s)$. We write r_A instead of $r_A(s, a_1^i, a_2^j)$
 414 when there is no risk of confusion. We define r_B and r_C similarly and normalize the revenues:
 415 $r_A + r_B + r_C = 1$.

416 To compute these values, we define ‘‘attractiveness’’. The attractiveness of a pool is its revenue
 417 divided by the total computing power of its miners.

418 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
 419 $\alpha' = i\epsilon$ of the total network computing power to attack B and is receiving a corresponding
 420 fraction of B ’s revenue while not contributing to it. Therefore the attractiveness of pool B will be
 421 equal to: $attr_B = \frac{r_B}{\beta + \alpha'}$. Similarly we have $attr_A = \frac{r_A}{\alpha + \beta'}$, where $\beta' = j\epsilon$.

422 Now consider the sources for pool A ’s revenue. It either comes from A ’s own mining process or
 423 from collecting shares of B ’s revenue, therefore:

424
$$r_A = (\alpha - \alpha') + \alpha' \times attr_B,$$

425 and similarly $r_B = (\beta - \beta') + \beta' \times attr_A$. The previous four equations provide us with a
 426 system of linear equations which we can solve to obtain the values of $r_A, r_B, attr_A$ and $attr_B$.
 427 Since a fraction $\alpha' + \beta'$ of total computation power is used on attacking other pools, we have:
 428 $attr_C = \frac{1}{1 - \alpha' - \beta'}$.

429 ■ *Game transitions (δ).* Miners migrate between pools and a pool gains or loses mining power
 430 based on its attractiveness. If a pool is the most attractive option among the two, it gains ϵ new
 431 mining power with probability $\frac{2}{3}$, retains its current power with probability $\frac{1}{6}$ and loses ϵ power
 432 with probability $\frac{1}{6}$. On the other hand a pool that is not the most attractive option loses ϵ power
 433 with probability $\frac{2}{3}$, retains its current power with probability $\frac{1}{6}$ and attracts ϵ new mining power
 434 with probability $\frac{1}{6}$. These values were chosen for the purpose of demonstration of our algorithm
 435 and our implementation results. In practice, one can obtain realistic probabilities experimentally.

436 ■ *Ergodicity.* The game is ergodic because for each two states $s = (s_1, s_2)$ and $s' = (s'_1, s'_2)$ where
 437 $|s_1 - s'_1| \leq 1$ and $|s_2 - s'_2| \leq 1$, there is at least $\frac{1}{36}$ probability of going from s to s' no matter
 438 what choices the players make.

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

441 **5.2 Zero-confirmation Double-spending**

442 Nowadays, Bitcoin is increasingly used in ‘‘fast payments’’ such as online services, ATM withdrawals
 443 and vending machines [22], where the payment is followed by fast delivery of goods. While the
 444 blockchain consensus is appropriate for slow payments, it requires tens of minutes to confirm a

445 transaction and is therefore inappropriate for fast payments. We consider a transaction confirmed
446 when it is added to the blockchain and several blocks are added after it. This mechanism is essential
447 for the detection of double-spending attacks in which an adversary attempts to use some of her coins
448 for two or more payments. However, even in the absence of a confirmation, it is far from trivial to
449 perform a double-spending attack. In a double spending attack, the attacker publishes two transactions
450 that consume the same input. The attack is successful only if the victim node received one transaction
451 and provided the goods before he became aware of the other, but eventually the latter was added to
452 the blockchain. In an ideal world the attacker can increase his odds by broadcasting one transaction
453 directly to the victim and the other at a far apart location, while on the other hand the victim can
454 defend itself by deploying several nodes in the network in *strategic* locations. In the real world,
455 however, the full topology of the network is never known to either of the parties. Nevertheless, based
456 on history and network statistics one can estimate the odds of a successful attack given the current
457 state of the network [11].

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

467 **Our modeling.** We aim to analyze the worst case long run loss of the victim. In our model we
468 abstract the network topology state and consider only the odds of successful double spending. We
469 consider a scenario where the victim's honest customers typically purchase goods worth 10 units per
470 round. In every round, the victim decides on a policy for accepting fast payment, and the attacker,
471 concurrently, unaware of the victim's policy, has to decide the size of the attack. After every round,
472 the victim decides if he wants to do a thorough change in the network topology. If he decides on
473 a change, then the next state is chosen uniformly from all possible states (this represents the fact
474 that neither players has full knowledge on the topology). If he decides to make no change, then the
475 network state might still change, due to the dynamic nature of the network. In this case the next state
476 is with high probability either the current state, or a state which is slightly better or slightly worse for
477 the victim, but with low probability the state changes completely to an arbitrary state in the network
478 (as sometimes small changes in the topology have big impact). The rewards stem from the outcome
479 of each round in the following way: The payment is the sum of the honest customer purchases and
480 the payment of the attacker (if it gets into the blockchain). The reward is the payment minus some
481 penalty in case the victim has decided to wait for a confirmation. The fact that the network state is
482 constantly changing makes our model ergodic. A proof and more details of the following Theorem
483 are provided in Appendix C.1.

484 ► **Theorem 5.** *Consider a seller and an attacker in the zero-confirmation double spending problem.*
485 *The seller can ensure profit of at least v on average per round, where v is the value of the corresponding*
486 *CMPG.*

487 5.3 Proof of Stake Pool Attack

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

494 **Our modeling.** We assume a setting similar to that of Section 5.1, where there are two opponent
 495 pools A and B , and the rest of the network consists of honest pools who sign every block that arrives
 496 on time. The states of the game are the stakes of each pool, namely α for pool A and β for pool B .
 497 In every round, with probability $1 - (\alpha + \beta)$ neither of the pools is elected to mine a block, and
 498 no decisions are made. Otherwise, with probability $\frac{\alpha}{\alpha + \beta}$ pool A is elected and otherwise pool B is
 499 elected. When a pool is elected, the other pool decides whether to sign and broadcast the resulting
 500 block or not. In addition the network state and connectivity induce a distribution over the fraction of
 501 honest miners that receive the block. If the block is accepted, then its creator is rewarded with mining
 502 fees, and the other pool will get its signing fees only if it signed the block. A proof and more details
 503 of the following Theorem are provided in Appendix C.2

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

510 6 Implementation and Experimental Results

511 In this section we present our implementation details and experimental results. The code is available
 512 at <http://ist.ac.at/~akafshda/concur2018>.

513 6.1 Implementation Challenges

514 We have implemented the strategy-iteration algorithm for ergodic games (see Appendix A for pseudo-
 515 code and more details). To the best of our knowledge, this is the first implementation of this algorithm.
 516 The straightforward implementation of the strategy-iteration algorithm for ergodic games has two
 517 practical problems, which we describe below.

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

526 Our solution for the above two problems are as follows:

- 527 1. *Stopping criteria.* We first observe that the value sequence which is obtained converges from
 528 below to the value of the game. In other words, the value sequence provide a lower bound to the
 529 lower value of the game. Hence we consider a symmetric version which is the strategy-iteration
 530 algorithm for player 2, and run each iteration of the two algorithms in sequence. The version for
 531 player 2 provides a lower bound on the lower value for player 2, and thus from that we can obtain
 532 an upper bound on the upper value of player 1. Since the upper and lower values coincide, we
 533 thus have both an upper and lower bound on the values, and once the difference is smaller than
 534 $\epsilon > 0$, then the algorithm has correctly approximated the value within ϵ and can stop and return
 535 the value and the strategy obtained as approximation.
- 536 2. *Numerical precision.* For numerical precision, instead of obtaining the results from the linear
 537 program, we obtain from the linear program the set of *tight* and *slack* constraints, where the tight
 538 constraints represent the constraints where equality is obtained, and the other constraints are
 539 slack ones. From the tight constraints, which are equalities, we obtain the result using Gaussian
 540 elimination, which provides more precise values to the solution. We also provide other heuristics,

541 such as adding the remaining probability to the greatest probability action, and obtain similar
542 results on convergence.

543 6.2 Experimental Results

544 We provide experimental results for all games in Section 5. We show number of transitions in the
545 game (#T), number of states in the game, the running time and number of strategy iterations (#SI) for
546 every scenario.

| #T | States | #SI | Time(s) | #T | States | #SI | Time(s) | #T | States | #SI | Time(s) |
|---------|--------|-----|---------|--------|--------|-----|---------|--------|--------|-----|---------|
| 17050 | 100 | 4 | 69 | 19940 | 100 | 2 | 426 | 6076 | 99 | 18 | 471 |
| 56252 | 196 | 2 | 291 | 40040 | 200 | 2 | 800 | 20956 | 275 | 8 | 1338 |
| 135252 | 289 | 2 | 389 | 60140 | 300 | 2 | 1141 | 31744 | 396 | 9 | 2520 |
| 236000 | 400 | 2 | 1059 | 80240 | 400 | 2 | 1586 | 44764 | 539 | 4 | 1073 |
| 331816 | 484 | 2 | 3880 | 100340 | 500 | 2 | 2069 | 77500 | 891 | 16 | 22125 |
| 508032 | 576 | 2 | 6273 | 120440 | 600 | 2 | 1253 | 119164 | 1331 | 27 | 32636 |
| 720954 | 676 | 2 | 17014 | 140540 | 700 | 2 | 2999 | 169756 | 1859 | 10 | 31597 |
| 966281 | 784 | 2 | 53103 | 160640 | 800 | 2 | 3496 | 262384 | 2816 | 12 | 89599 |
| 1269450 | 900 | 2 | 100435 | 180740 | 900 | 2 | 3917 | | | | |

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

547 Note that #SI is not monotone in the number of states. Intuitively the number of needed iterations
548 depends on the extent in which easy locally optimal strategies are also globally optimal. In addition
549 the strategy iteration algorithm starts with an arbitrary random strategy, and hence the number of
550 iterations also depends on the initial strategy. However, it is worthy to note that in all cases the number
551 of iterations required is quite small. We also note that since the number of iterations is small, the
552 crucial computational step is every iteration, where many linear-programming problems are solved.

553 **Outputs of the algorithm.** The outputs provided the following results:

- 554 ■ For the block withholding pool attack game, the algorithm could guarantee a mean-payoff of
555 0.49 for the victim pool. In absence of an attacker the pool becomes the most attractive option
556 for miners and grows to maximum possible size with probability 1, hence if there is no other
557 pool the mean-payoff will be 1. Also, if there are two pools A and B with hash powers α and
558 β respectively, and they decide not to attack each other, then they will both become the most
559 attractive option and will grow with the same rate, leading to a mean-payoff of $\alpha + \frac{1-\alpha-\beta}{2}$ for A
560 and $\beta + \frac{1-\alpha-\beta}{2}$ for B .
- 561 ■ For the zero-confirmation double-spending game, the algorithm verified that the seller is guaran-
562 teed to maintain at least half of her revenue, i.e., in presence of a malicious attacker, the value for
563 the seller converges to 5 as the number of states increase, while it is 10 in absence of it.
- 564 ■ For the proof of stake pool attack game, by increasing the number of states, i.e., by refining the
565 discretization, the guaranteed value (game value) decreases and tends to zero. In absence of an
566 attacker, a pool A can achieve an expected payoff of $11s_A$ at a turn where s_A is the stake it holds.
567 This is because it earns an average of $10s_A$ from mining fees and s_A from signing. In this case,
568 since the pool becomes the most attractive option, it gains miners and reaches a stake of 1, leading
569 to a mean-payoff of 11.

570 For the exact details see Appendix C. Our algorithm also finds strategies that achieve these values.

571

572 7 Related Work

573 *Basic bitcoin security.* The first security analysis of the Bitcoin protocol was done by Nakamoto [36]
574 who showed the resilience of the blockchain protocol against a double-spending attack. His analysis
575 was later corrected by Rosenfeld [41] who showed that the use of probabilistic arguments in the
576 original analysis was not sound. Rosenfeld's analysis gives different numerical results, but still

577 certifies the original security properties. Recently Sompolinsky and Zohar [45] further refined the
 578 analysis by considering the fact that the attacker can observe the possible states of the blockchain
 579 before choosing to attack, and thus he can increase his utility by choosing the right time to attack.

580 *Pools attack.* The danger of a block withholding attack is as old as Bitcoin pools. The attack was
 581 described by Rosenfeld [40] as early as 2011, as pools were becoming a dominant player in the
 582 Bitcoin world. While it was obvious that a pool is vulnerable to a malicious attacker, Eyal [24]
 583 showed that in some circumstances a pool can benefit by attacking another pool, and thus pool mining
 584 is vulnerable also in the presence of rational attackers. However, the analysis only considered the
 585 short term, i.e., the profit that the pool can get only in the short period after the attack. Laszka et
 586 al. [34] studied the long term impact of pools attack. In their framework miners are allowed to migrate
 587 from one pool to another. They analyzed the steady equilibrium in which the size of the pools become
 588 stable (although there is no guarantee that the game will converge to such a scenario). Our framework
 589 is the first to allow analysis of long term impacts without convergence assumptions.

590 *Zero-confirmation double-spending.* Zero-confirmation double-spending was experimentally analyzed
 591 by Karame et al. [31] who gave numerical figures for the odds of successful double spending for
 592 different network states. However, their analysis did not consider the fact that the victim may
 593 change his connectivity state. Our work is the first analysis framework for the long term impact of
 594 zero-confirmation double-spending.

595 *Stateful analysis.* A stateful analysis of blockchain attacks was done by Sapirshtein et al. [43] and by
 596 Sompolinsky and Zohar [45]. In their analysis the different states of the blockchain were taken into
 597 account during the attack. The analysis was done using MDPs (a single player game, where only one
 598 player makes the choices) in which only the attacker decides on his actions and the victim follows a
 599 predefined protocol. A recent work [16] has also considered abstraction-refinement for finite-horizon
 600 games in the context of smart contracts. However, it neither considers long-term behavior, nor
 601 mean-payoff objectives, nor can it model attacks such as double-spending and interactions between
 602 pools (see Appendix B for more details).

603 *Quantitative verification with mean-payoff games.* The mean-payoff games problem has been studied
 604 extensively as a theoretical problem in various models [38, 39]. The mean-payoff games problem
 605 has also been studied in the context of verification and synthesis for performance related issues
 606 [12, 15, 20, 7, 6, 26] (see Section 4.1 for more details). However all these works focus on turn-based
 607 games, and none of them consider concurrent games. To the best of our knowledge concurrent
 608 mean-payoff games have not been studied in the setting of security that we consider, where the
 609 quantitative objective is as crucial as safety critical issues. Practical implementation of algorithms for
 610 ergodic CMPGs do not exist in the literature.

611 *Formal methods in security.* There is a huge body of work on program analysis for security (see [42, 1]
 612 for surveys). Formal methods are used to create safe programming languages (e.g., [27, 48, 42])
 613 and to define new logics that can express security properties (e.g., [14, 4, 3]). They are also used to
 614 automatically verify security and cryptographic protocols, e.g., [2, 10] and [5] for a survey. However,
 615 all of these works aimed to formalize qualitative properties such as privacy violation and information
 616 leakage. The works of [32, 21] consider analysis of security protocols with turn-based games and
 617 qualitative properties. To our knowledge, our framework is the first attempt to use concurrent
 618 mean-payoff games as a tool for reasoning about economic effects of attacks in crypto-currencies.

619 **8 Conclusion and Future Work**

620 In this work we considered concurrent mean-payoff games, and in particular the subclass of ergodic
 621 games, to analyze attacks on crypto-currencies. There are several interesting directions to pursue:
 622 First, various notions of rationality are relevant to analyze games where the attacker is rational, rather
 623 than malicious, and aims to maximize his own utility instead of minimizing the defender's utility
 624 (e.g., secure-equilibria [17] or other related notions). Second, we consider two-player games, and the
 625 extension to multi-player games to model crypto-currency attacks is another interesting problem.

626 — **References** —

- 627 1 M. Abadi. Software security: A formal perspective - (notes for a talk). In *FM 2012: Formal*
628 *Methods - 18th International Symposium, Paris, France, August 27-31, 2012. Proceedings*, pages
629 1–5, 2012.
- 630 2 M. Abadi and P. Rogaway. Reconciling two views of cryptography. In *Proceedings of the IFIP*
631 *International Conference on Theoretical Computer Science*, pages 3–22. Springer, 2000.
- 632 3 O. Arden, J. Liu, and A.C. Myers. Flow-limited authorization. In *IEEE 28th Computer Security*
633 *Foundations Symposium, CSF 2015, Verona, Italy, 13-17 July, 2015*, pages 569–583, 2015.
- 634 4 O. Arden and A. Myers. A calculus for flow-limited authorization. In *29th IEEE Symp. on Com-*
635 *puter Security Foundations (CSF)*, 2016.
- 636 5 M. Avalle, A. Pironti, and R. Sisto. Formal verification of security protocol implementations: a
637 survey. *Formal Aspects of Computing*, 26(1):99–123, 2014.
- 638 6 C. Baier, C. Dubslaff, J. Klein, S. Klüppelholz, and S Wunderlich. Probabilistic model checking
639 for energy-utility analysis. In *Horizons of the Mind. A Tribute to Prakash Panangaden - Essays*
640 *Dedicated to Prakash Panangaden on the Occasion of His 60th Birthday*, pages 96–123, 2014.
- 641 7 C. Baier, S. Klüppelholz, H. de Meer, F. Niedermeier, and S. Wunderlich. Greener bits: Formal
642 analysis of demand response. In *Automated Technology for Verification and Analysis - 14th Interna-*
643 *tional Symposium, ATVA 2016, Chiba, Japan, October 17-20, 2016, Proceedings*, pages 323–339,
644 2016.
- 645 8 Bitcoin Wiki. Comparison of mining pools, 2017. URL: [http://en.bitcoin.it/](http://en.bitcoin.it/Comparison_of_mining_pools)
646 [Comparison_of_mining_pools](http://en.bitcoin.it/Comparison_of_mining_pools).
- 647 9 D. Blackwell and T.S. Ferguson. The big match. *AMS*, 39:159–163, 1968.
- 648 10 B. Blanchet and A. Chaudhuri. Automated formal analysis of a protocol for secure file sharing on
649 untrusted storage. In *IEEE Symposium on Security and Privacy*, 2008.
- 650 11 blockcypher.com. Confidence factor, 2017. URL: [http://dev.blockcypher.com/](http://dev.blockcypher.com/#confidence-factor)
651 [#confidence-factor](http://dev.blockcypher.com/#confidence-factor).
- 652 12 R. Bloem, K. Chatterjee, T.A. Henzinger, and B. Jobstmann. Better quality in synthesis through
653 quantitative objectives. In *CAV 2009*, pages 140–156, 2009. URL: [http://dx.doi.org/10.](http://dx.doi.org/10.1007/978-3-642-02658-4_14)
654 [1007/978-3-642-02658-4_14](http://dx.doi.org/10.1007/978-3-642-02658-4_14), doi:10.1007/978-3-642-02658-4_14.
- 655 13 J. Bonneau, A. Miller, J. Clark, A. Narayanan, J.A. Kroll, and E.W. Felten. Sok: Research per-
656 spectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security*
657 *and Privacy*, pages 104–121. IEEE, 2015.
- 658 14 M. Burrows, M. Abadi, and R.M. Needham. A logic of authentication. In *Proceedings of the Royal*
659 *Society of London A: Mathematical, Physical and Engineering Sciences*, pages 233–271. The Royal
660 Society, 1989.
- 661 15 P. Cerný, K. Chatterjee, T.A. Henzinger, A. Radhakrishna, and R. Singh. Quantitative synthesis for
662 concurrent programs. In *CAV 2011*, pages 243–259, 2011. URL: [http://dx.doi.org/10.](http://dx.doi.org/10.1007/978-3-642-22110-1_20)
663 [1007/978-3-642-22110-1_20](http://dx.doi.org/10.1007/978-3-642-22110-1_20), doi:10.1007/978-3-642-22110-1_20.
- 664 16 K. Chatterjee, A.K. Goharshady, and Y. Velner. Quantitative analysis of smart contracts. In
665 *European Symposium on Programming (arXiv:1801.03367)*, 2018.
- 666 17 K. Chatterjee, T.A. Henzinger, and M. Jurdzinski. Games with secure equilibria. In *19th IEEE Sym-*
667 *posium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings*,
668 pages 160–169, 2004.
- 669 18 K. Chatterjee and R. Ibsen-Jensen. The complexity of ergodic mean-payoff games. In *ICALP II*
670 *2014*, pages 122–133, 2014.
- 671 19 K. Chatterjee, R. Majumdar, and T. A. Henzinger. Stochastic limit-average games are in EXPTIME.
672 *Int. J. Game Theory*, 37(2):219–234, 2008.
- 673 20 K. Chatterjee, A. Pavlogiannis, and Y. Velner. Quantitative interprocedural analysis. In *POPL*
674 *2015*, pages 539–551, 2015. URL: <http://doi.acm.org/10.1145/2676726.2676968>,
675 doi:10.1145/2676726.2676968.
- 676 21 K. Chatterjee and V. Raman. Assume-guarantee synthesis for digital contract signing.
677 *Formal Asp. Comput.*, 26(4):825–859, 2014. URL: [http://dx.doi.org/10.1007/](http://dx.doi.org/10.1007/s00165-013-0283-6)
678 [s00165-013-0283-6](http://dx.doi.org/10.1007/s00165-013-0283-6), doi:10.1007/s00165-013-0283-6.

- 679 **22** CNN Money. Bitcoin's uncertain future as currency, 2011. URL: [http://money.cnn.com/
680 video/technology/2011/07/18/t_bitcoin_currency.cnnmoney/](http://money.cnn.com/video/technology/2011/07/18/t_bitcoin_currency.cnnmoney/).
- 681 **23** coinmarketcap.com. Crypto-currency market capitalizations, 2017. URL: [http://
682 coinmarketcap.com/](http://coinmarketcap.com/).
- 683 **24** I. Eyal. The miner's dilemma. In *2015 IEEE Symposium on Security and Privacy*, pages 89–103.
684 IEEE, 2015.
- 685 **25** I. Eyal and E.G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Crypto-
686 graphy and Data Security*, 2014.
- 687 **26** V. Forejt, M. Z. Kwiatkowska, and D. Parker. Pareto curves for probabilistic model checking. In
688 *Automated Technology for Verification and Analysis - 10th International Symposium, ATVA 2012,
689 Thiruvananthapuram, India, October 3-6, 2012. Proceedings*, pages 317–332, 2012.
- 690 **27** A.P. Fuchs, A. Chaudhuri, and J.S. Foster. Scandroid: Automated security certification of android.
691 *Technical report*, 2009.
- 692 **28** D. Gillette. Stochastic games with zero stop probabilities. In *CTG*, pages 179–188. Princeton
693 University Press, 1957.
- 694 **29** K. A. Hansen, M. Koucký, N. Lauritzen, P. B. Miltersen, and E. P. Tsigaridas. Exact algorithms for
695 solving stochastic games: extended abstract. In *STOC*, pages 205–214, 2011.
- 696 **30** A.J. Hoffman and R.M. Karp. On nonterminating stochastic games. *Management Sciences*,
697 12(5):359–370, 1966.
- 698 **31** G. Karame, E. Androulaki, and S. Capkun. Two bitcoins at the price of one? double-spending
699 attacks on fast payments in bitcoin. *IACR Cryptology ePrint Archive*, 2012:248, 2012.
- 700 **32** S. Kremer and J.F. Raskin. A game-based verification of non-repudiation and fair exchange proto-
701 cols. *Journal of Computer Security*, 2003.
- 702 **33** J. Kwon. Tendermint: Consensus without mining, 2015. URL: [https://blog.ethereum.
703 org/2015/08/01/introducing-casper-friendly-ghost/](https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/).
- 704 **34** A. Laszka, B. Johnson, and J. Grossklags. When bitcoin mining pools run dry. In *International
705 Conference on Financial Cryptography and Data Security*, pages 63–77. Springer, 2015.
- 706 **35** J.F. Mertens and A. Neyman. Stochastic games. *IJGT*, 10:53–66, 1981.
- 707 **36** S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.
- 708 **37** NxtCommunity. Nxt whitepaper, 2014. URL: [http://bravenewcoin.com/assets/
709 Whitepapers/NxtWhitepaper-v122-rev4.pdf](http://bravenewcoin.com/assets/Whitepapers/NxtWhitepaper-v122-rev4.pdf).
- 710 **38** A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL'89*, pages 179–190. ACM
711 Press, 1989.
- 712 **39** P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *IEEE Transactions on
713 Control Theory*, 77:81–98, 1989.
- 714 **40** M. Rosenfeld. Analysis of bitcoin pooled mining reward systems. *arXiv*, 2011.
- 715 **41** M. Rosenfeld. Analysis of hashrate-based double spending. *CoRR*, abs/1402.2009, 2014.
- 716 **42** A. Sabelfeld and A.C. Myers. Language-based information-flow security. *IEEE Journal on Selected
717 Areas in Communications*, 21(1):5–19, 2003.
- 718 **43** A. Sapirshstein, Y. Sompolinsky, and A. Zohar. Optimal selfish mining strategies in bitcoin. *arXiv
719 preprint arXiv:1507.06183*, 2015.
- 720 **44** L.S. Shapley. Stochastic games. *PNAS*, 39:1095–1100, 1953.
- 721 **45** Y. Sompolinsky and A. Zohar. Bitcoin's security model revisited. *CoRR*, abs/1605.09193, 2016.
- 722 **46** M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state systems. In *FOCS'85*,
723 pages 327–338. IEEE Computer Society Press, 1985.
- 724 **47** V. Zamfir. Introducing casper, the friendly ghost, 2015. URL: [https://blog.ethereum.
725 org/2015/08/01/introducing-casper-friendly-ghost/](https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/).
- 726 **48** D. Zhang, Y. Wang, G. E. Suh, and A. C. Myers. A hardware design language for timing-sensitive
727 information-flow security. In *Proceedings of the Twentieth International Conference on Architec-
728 tural Support for Programming Languages and Operating Systems, ASPLOS '15, Istanbul, Turkey,
729 March 14-18, 2015*, pages 503–516, 2015.

A The Hoffman-Karp Strategy-iteration Algorithm

730

731 For an ergodic CMPG G and a state t , the basic informal description of the algorithm is as follows.
 732 In every iteration i , the algorithm considers a stationary strategy σ_1^i , and then improves the strategy
 733 locally as follows: first it computes the *potential* $v_s^{\sigma_1^i}$ (described below) given σ_1^i , and then for every
 734 state s , the algorithm locally computes an arbitrary optimal distribution at s to improve the potential.
 735 The intuitive description of the potential is as follows: Fix the specific state t as the target state (where
 736 the potential must be 0); and given a stationary strategy σ , consider a modified reward function that
 737 assigns the original reward minus the value ensured by σ . Then the potential for every state s other
 738 than the specified state t is the expected sum of rewards under the modified reward function for the
 739 random walk from s to t . The local improvement step is obtained as a solution of a matrix game with
 740 the potentials. The formal description of the algorithm is given in Figure 4, and the formal definition
 741 of the expected one-step reward ExpRew and one-step function OneSt is below.

742 *Notations:* ExpRew and OneSt . The expected one-step reward $\text{ExpRew}(s, \sigma_1^i, a_2)$ for a stationary
 743 strategy σ_1^i for Player 1, that specifies a distribution $\sigma_1^i(s)$ for every state, and an action $a_2 \in \Gamma_2(s)$,
 744 is as follows:

$$745 \quad \text{ExpRew}(s, \sigma_1^i, a_2) = \sum_{a_1 \in \Gamma_1(s)} R(s, a_1, a_2) \cdot \sigma_1^i(s)(a_1) .$$

746 Similarly, we will also use the following notation:

$$747 \quad \delta(s, \sigma_1^i, a_2)(s') = \sum_{a_1 \in \Gamma_1(s)} \delta(s, a_1, a_2)(s') \cdot \sigma_1^i(s)(a_1) .$$

748 For notional convenience, given a vector $x = (x_i)_{i \in S}$, a state s and a pair of distributions $d_1 \in$
 749 $\mathcal{D}(\Gamma_1(s))$ and $d_2 \in \mathcal{D}(\Gamma_2(s))$, we let $\text{OneSt}(x, d_1, d_2, s)$ be

$$750 \quad \text{OneSt}(x, d_1, d_2, s) = \sum_{\substack{a_1 \in \Gamma_1(s) \\ a_2 \in \Gamma_2(s) \\ s' \in S}} d_1(a_1) \cdot d_2(a_2) \cdot \delta(s, a_1, a_2)(s') \cdot x_{s'} .$$

751 Also, given a vector $x = (x_i)_{i \in S}$, a state s and a stationary strategy profile $\sigma = (\sigma_1, \sigma_2)$, we will let
 752 $\text{OneSt}(x, \sigma, s)$ be

$$753 \quad \text{OneSt}(x, \sigma, s) = \text{OneSt}(x, \sigma_1(s), \sigma_2(s), s)$$

754 **Computation of every iteration.** The computation of every iteration is as follows. The computation
 755 of the unique solution g^i and $(v_s^i)_{s \in S}$ is obtained in polynomial time using linear programming.
 756 The fact that the solution is unique follows from the fact that once a strategy for Player 1 is fixed,
 757 we obtain an MDP for Player 2, and then the MDP solution is unique. The value sequence $(g^i)_{i \geq 1}$
 758 obtained by the Hoffman-Karp algorithm converges to the value g of the game [30].

B Comparison with other game-theoretic works

759

760 Previous works [16, 13] consider either one-shot or finite-horizon games for security analysis. In
 761 contrast, the main differences of our work are as follows:

- 762 ■ Finite-horizon (or bounded-horizon) games can be reduced to one-shot games with an exponential
 763 blow up in the number of strategies. Thus one-shot and finite-horizon games are conceptually
 764 similar, though there are computational complexity differences. In contrast to finite-horizon
 765 games, we consider infinite-horizon games, which is conceptually different from finite-horizon
 766 games and there is no reduction (even with a blow up) to finite-horizon or one-shot games.

Function HoffmanKarp(G, t)

```

Let  $\sigma_1^1$  be a Player-1 stationary strategy;
for ( $i \in \mathbb{Z}_+$ ) do
  Compute  $g^i, (v_s^i)_{s \in S}$  as the unique solution of
    
$$g^i + v_s^i = \min_{a_2 \in \Gamma_2(s)} (\text{ExpRew}(s, \sigma_1^i, a_2) + \text{OneSt}(v^i, \sigma_1^i(s), a_2, s))$$

    
$$\forall s \in S$$

    
$$v_t^i = 0;$$

    for ( $s \in S$ ) do
      Let  $M_s$  be the matrix game defined as follows;
      for ( $a_1 \in \Gamma_1(s)$  and  $a_2 \in \Gamma_2(s)$ ) do
         $P_s[a_1, a_2] := \text{OneSt}(v^i, a_1, a_2, s);$ 
         $M_s[a_1, a_2] := R(s, a_1, a_2) + P_s[a_1, a_2];$ 
      if ( $\sigma_1^i(s)$  is an optimal distribution for  $M_s$ ) then
         $\sigma_1^{i+1}(s) := \sigma_1^i(s);$ 
      else
         $\sigma_1^{i+1}(s) := \text{Optimal distribution over } \Gamma_1(s) \text{ for } M_s;$ 
    if ( $\sigma_1^{i+1} = \sigma_1^i$ ) then
      return  $\sigma_1^i;$ 

```

■ **Figure 4** Strategy-iteration algorithm for solving ergodic games

- 767 ■ In finite-horizon games for cryptocurrency, the focus is on abstraction-refinement [16]. In contrast,
 768 we consider a special class of games (ergodic games) and use algorithmic approaches for finding
 769 their values.
- 770 ■ In this work we consider attacks that are inherent in the Blockchain, such as double-spending and
 771 pool-attacks. Previous works do not consider the analysis of such attacks.

772 **C Formal Modeling of Problems as Concurrent Games**

773 **C.1 Formal Modeling of Zero-confirmation Double-spending**

- 774 ■ *Game states.* Each state of the game corresponds to a probability of success for double spending
 775 attack which is an abstraction of the network topology.
 776 We discretized the game into $n + 1$ states and set

777
$$S = \{0, 1, \dots, n\}.$$

778 The state 0 is called a shuffling state. Each other state i corresponds to a double spending success
 779 probability of $p_i = 0.1 + \frac{(i-1) \times 0.4}{n}$ for $1 \leq i \leq n$. Player 1 is the seller and player 2 is the
 780 malicious buyer.

- 781 ■ *Actions at each state.* The shuffling state, 0, corresponds to the seller deciding to disconnect
 782 and reconnect to the network so as to randomly obtain one of the other states. Therefore each
 783 player has only one action, i.e. no choice, in this state. We denote these actions by a_1^0 and a_2^0
 784 respectively.

785 At each other state the seller can decide to disconnect from the network and then reconnect to
 786 it. Moreover he can choose whether to require a confirmation and wait for it, so the seller has 4
 787 possible actions. We denote these actions by a_1^i as in the following table:

| action | reconnect | confirmation |
|---------|-----------|--------------|
| a_1^0 | No | No |
| a_1^1 | Yes | No |
| a_1^2 | No | Yes |
| a_1^3 | Yes | Yes |

The malicious buyer can decide how much double spending to attempt. He can attempt between 1 and 20 units of double spending. We denote the action of attempting d units of double spending as a_2^d .

- *Game transitions.* If the game is in the shuffling state, the next state will be one of the other n states and all of them are equally likely, i.e.

$$\delta(0, a_1^0, a_2^0)(i) = \frac{1}{n},$$

for $1 \leq i \leq n$.

Otherwise, if the seller decides to reset his connection to the network, the game will be transitioned to the shuffling state with probability 1, no matter what choice was made by the buyer. More formally,

$$\delta(s, a_1^1, a_2)(0) = \delta(s, a_1^3, a_2)(0) = 1,$$

for all $s \in S \setminus \{0\}$ and $a_2 \in \Gamma_2(s)$.

Otherwise, if the seller decides to wait for a confirmation, then the attack will be unsuccessful and if he does not wait for a confirmation, then the current state defines the odds of a successful attack. So the attack will succeed with probability

$$p_a(s, a_1) = \begin{cases} p_s & a_1 \in \{a_1^0, a_1^1\} \\ 0 & a_1 \in \{a_1^2, a_1^3\} \end{cases}.$$

We consider two cases. If the attack is successful, the game transitions to state n , i.e. the state with the highest probability of double-spending success,

$$\delta_a(s, a_1, a_2)(n) = p_a(s, a_1),$$

for $s \in S \setminus \{0\}$, $a_1 \in \{a_1^0, a_1^2\}$ and $a_2 \in \Gamma_2(s)$. Intuitively, this is because if the attacker was successful once, he can repeat the attack.

If the attack fails, then if the game is at state s , it goes to each of the states $s - 1$, s and $s + 1$ (if they exist and are non-zero) with equal probability. This captures small changes in the topology of the network that can be caused by factors that are not parties to the game, like other people reconnecting to the network. Also the game will transition to the shuffling state with a small probability p_{dc} . This models the natural loss of connection that may occur in the network and cause the seller to reconnect even though he did not intentionally decide to do so. In the implementation we set $p_{dc} = 0.001$. More formally by letting $N_s = \{s - 1, s, s + 1\} \cap \{1, 2, \dots, n\}$, we have

$$\delta_b(s, a_1, a_2)(0) = p_{dc}(1 - p_a(s, a_1)),$$

$$\delta_b(s, a_1, a_2)(s') = \frac{1 - p_a(s, a_1)}{|N_s|} \times (1 - p_{dc}),$$

where $s \in S \setminus \{0\}$, $s' \in N_s$, $a_1 \in \{a_1^0, a_1^2\}$ and $a_2 \in \Gamma_2(s)$.

Finally we set the probability of transitioning to a state s' as the sum of the two probabilities obtained in the cases above:

$$\delta(s, a_1, a_2)(s') = \delta_a(s, a_1, a_2)(s') + \delta_b(s, a_1, a_2)(s')$$

for $s, s' \in S \setminus \{0\}$, $a_1 \in \{a_1^0, a_1^2\}$ and $a_2 \in \Gamma_2(s)$.

825 ■ *Rewards.* The rewards model net income (profit) of the seller. Transitions from the shuffling state
 826 carry a reward of zero, since the seller is unable to sell any goods while his connection is being
 827 reset. Formally,

$$828 \quad R(0, a_1^0, a_2^0) = 0.$$

829 Assuming that the seller has a profit ratio of p . We model the rewards to capture his profit. In the
 830 implementation we have set $p = 0.5$. Recall that the malicious buyer, when choosing action a_2^d ,
 831 is trying to double-spend an amount d , $1 \leq d \leq 20$ and that other buyers are interested in buying
 832 10 units of goods from the seller.

833 Again we consider two cases. If the double spending attack is successful, this will yield to a total
 834 payoff of $-d(1 - p)$ for the seller while an unsuccessful attack gives him a profit of dp . So we
 835 can set

$$836 \quad R_1(s, a_1, a_2^d) = dp(1 - p_a(s, a_1)) - d(1 - p)p_a(s, a_1)$$

837 for $s \in S \setminus \{0\}$, $a_1 \in \Gamma_1(s)$ and $1 \leq d \leq 20$.

838 Now we focus on the profit of selling to other (non-malicious) buyers. If the seller decides to
 839 wait for a confirmation, he will not be able to serve a fraction f of his other customers, who are
 840 not willing to wait. We have set $f = 0.5$ in the implementation. So he gets a total revenue of
 841 $10p(1 - f)$ from his other customers. On the other hand, if he does not wait for a confirmation he
 842 will receive a payoff of $10p$, so

$$843 \quad R_2(s, a_1^0, a_2^d) = R_2(S, a_1^1, a_2^d) = 10p,$$

$$844 \quad R_2(s, a_1^2, a_2^d) = R_2(s, a_1^3, a_2^d) = 10p(1 - f),$$

846 for $s \in S \setminus \{0\}$ and $1 \leq d \leq 20$.

847 The final payoff is the sum of profits that the seller makes by selling to the malicious buyer and
 848 others, i.e.

$$849 \quad R(s, a_1, a_2) = R_1(s, a_1, a_2) + R_2(s, a_1, a_2)$$

850 for $s \in S \setminus \{0\}$, $a_1 \in \Gamma_1(s)$ and $a_2 \in \Gamma_2(s)$.

851 ■ *Ergodicity.* This game is ergodic. Starting with any state and strategy profile, the shuffling state,
 852 0, is visited infinitely often with probability 1. This is because any choice of actions by the
 853 two players at each turn would switch the game to the shuffling state with probability at least
 854 p_{dc} . Since the shuffling state is visited infinitely often, and since all other states have a non-zero
 855 probability of following the shuffling state, we conclude that every state in the game is visited
 856 infinitely often with probability 1 and hence the game is ergodic.

857 **Proof of Theorem 5.** We have already shown that the game is ergodic. The rest is obtained from the
 858 modeling and the determinacy result.

859 C.2 Formal Modeling of Proof of Stake Pool Attack

860 ■ *Game States.* We consider two pools, A and B and, as in the block withholding game, assume
 861 that any miner outside the two pools mines independently and set $\epsilon = \frac{1}{2n+1}$. We set $S \subseteq$
 862 $\{1, 2, \dots, n\} \times \{1, 2, \dots, n\} \times \{0, 0.01, 0.02, \dots, 1\}$, where each state in S is of the form of
 863 a 3-tuple like $s = (i, j, p)$ and corresponds to a situation in the game where pool A has a total
 864 mining power of $i\epsilon$, pool B has $j\epsilon$ and whenever a mined block is announced, the number of
 865 independent stake signatures that it receives is drawn from the Poisson distribution with parameter
 866 $(1 - i\epsilon - j\epsilon)p$. The intuition is that p is a measure of connectivity of the network and each
 867 miner sees the block and signs it with this probability. We are using Poisson distribution as a
 868 rough continuous approximation of the binomial distribution. In real life, the distribution can be
 869 obtained by trial and error on the network.

870 ■ *Actions at each state.* Each pool has two choices at each state: to sign a block mined by the other
871 pool, or to refrain from signing. We show these with a_1^s, a_1^r for pool A and a_2^s, a_2^r for B .

872 ■ *Rewards.* We consider pool A 's revenue as game rewards. Several cases should be considered:

- 873 1. If A is chosen to mine the next block and the mined block gets signed by a majority of stakes,
874 either including B or not, then A gets a mining reward of 10 units.;
- 875 2. If B is chosen to mine the next block and A opts to sign it then A gets a signing reward of $i\epsilon$,
876 i.e. the total signing reward for each block is 1 unit;
- 877 3. Similarly if an independent miner, or A itself for that matter, gets to mine the next block, A 's
878 revenue will be $i\epsilon$ units¹.

879 More concretely, we have $R_A = R_1 + R_2 + R_3$, where R_A is the revenue of pool A and
880 R_i corresponds to revenues from each of the parts above. Let CDF denote the cumulative
881 distribution function corresponding to the distribution mentioned above, then we have:

$$882 \quad R_1 = 10i\epsilon \times \begin{cases} 1 & i\epsilon \geq \frac{1}{2} \\ 1 & \text{B chooses } a_2^s \text{ and } i\epsilon + j\epsilon \geq \frac{1}{2} \\ 1 - CDF(\frac{1}{2} - i\epsilon) & \text{B chooses } a_2^r \text{ and } i\epsilon < \frac{1}{2} \\ 1 - CDF(\frac{1}{2} - i\epsilon - j\epsilon) & \text{B chooses } a_2^s \text{ and } i\epsilon + j\epsilon < \frac{1}{2} \end{cases},$$

883 The first case corresponds to the situation where A has enough stakes to sign his own block
884 with a majority. The second case is when A and B form a majority together and B has chosen
885 to sign A 's block. In the third case, A is not holding a majority and B is not signing the block,
886 so in order for A to get the block mining fees, a fraction of other miners holding at least $\frac{1}{2} - i\epsilon$
887 must sign the block. The fourth case captures the state where both A and B sign the block but
888 they do not form a majority.

$$889 \quad R_2 = j\epsilon \times \begin{cases} 0 & \text{A chooses } a_1^r \\ i\epsilon & \text{A chooses } a_1^s \end{cases},$$

$$890 \quad R_3 = i\epsilon(1 - j\epsilon).$$

891 Pool B 's revenue, R_B , can be defined similarly and will be used in the next part.

892 ■ *Game transitions.* The attractiveness of a pool is defined as its revenue divided by its stake, i.e.
893 $attr_A = \frac{R_A}{i\epsilon}$ and $attr_B = \frac{R_B}{j\epsilon}$. We do not consider the attractiveness of independent mining in
894 this game. A pool gains or loses mining stake based on its attractiveness. If it is the most attractive
895 of the two, it gains ϵ stake with probability $\frac{2}{3}$, retains its current stake with probability $\frac{1}{6}$ and loses
896 ϵ stake with probability $\frac{1}{6}$. Otherwise, it loses ϵ stake with probability $\frac{2}{3}$ and retains and gains
897 with probability $\frac{1}{6}$ each. This is very similar to the case in the block withholding game.

898 The value of p remains the same or switches to one of the neighboring values with equal probability.
899 This captures small changes in the network.

900 ■ *Ergodicity.* The argument for ergodicity is similar to the case of block withholding game.

901 **Proof of Theorem 6.** As above, ergodicity of the game is established in the exact same manner
902 as in the block withholding game. The rest follows straightforwardly from the modeling and the
903 determinacy result.

904 C.3 Details of Experimental Results

905 **Number of States.** The number of states in each of the experiments is determined as follows:

- 906 1. *Block withholding pool attack game.* In this game the number of states depends on the discretiza-
907 tion factor of the mining power. For example, for a discretization factor n , we say that pool A has
908 m units if its total mining power is m/n fraction of the entire computation power of the network.
909 Since we need to keep track of the mining power of A and B we have $O(n^2)$ states.

¹ We assume that A always signs blocks found by itself and independent miners.

910 **2. Zero-confirmation double-spending game.** Here the number of states are exactly the different
911 abstracted network states. For example if the minimal (maximal) odds for successful double-
912 spending are 30% (70%) and we consider a discretization factor of $1/n\%$, then we will have $40n$
913 states.

914 **3. Proof of stake pool attack game.** Here the number of states is dependent upon both the discretiza-
915 tion factor of mining stakes of the pools and the number of different abstracted network states. For
916 example, if we consider s network states and discretize mining stakes similar to Part 1 above, then
917 the game will have $O(sn^2)$ states to keep track of the stakes of both pools and the connectivity of
918 the network.

919 **Experiment Machine and Parameters.** We obtained the results using an AMD Dual-core
920 Opteron 885 (2.6 GHz) processor over Debian 3.2 OS with 32 GB of RAM and $\epsilon = 0.01$. The input
921 is an ergodic game as described in Section 5 and we are using Poisson distribution as the distribution
922 mentioned in the formal modeling of the proof of stake pool attack game.