D-02

公開鍵暗号を用いた P2P 型ネットワーク麻雀将棋

P2P Network Mahjong Shogi Using Public Key Cryptosystem

山崎 朋哉† 宮崎 修一‡
Tomoya Yamazaki Shuichi Miyazaki

1. はじめに

近年, インターネットを介して物理的に離れている相手と対戦が行えるゲームが広く普及している. このような対戦ゲームではサーバーを設置することが一般的である. しかし, 利用者が増加することでサーバーの負荷が増えてしまうという欠点から, P2P 方式のネットワーク上でゲームを行うことが望ましい場合があり, その設計が提案されている[1].

ネットワーク対戦ゲームでは互いがルールに則って進行していくことを前提としているが, 相手に知られることなく不正を行うプレイヤーが存在するかもしれない. 将棋やチェスのような完全情報ゲームでは, 相手に知られてはいけない秘匿情報が存在しないために, P2P 方式でもこのような不正が起こりにくい. しかし, トランプのようにプレイヤー全員が見ることができない山札や, 自分だけしか見ることができない手札が存在すると, P2P 方式ではこれらの情報管理が難しく, 特に他のプレイヤーに管理されることがないようにしなければならない. このような不正を排除するためには, 不正を監視するサーバーを設置することが一般的だが[2], 審判サーバー自体の信用性の疑惑や, 審判サーバー設置コストによるゲームの手軽さの阻害という問題が発生する. そこで, 本稿は, 不完全情報ゲームに対して, 審判サーバーを設置することなく, かつゲームを不正なく安全に進行するためのシステムを構築することを目的としている.

1 対 1 対戦ゲームは数多く存在し, トランプ等のゲームは主に互いの秘匿情報や山札の情報を盗み見るという不正が考えられるが, 互いの秘匿情報を参照する時に起こりうる不正を考える必要がない. 本稿では, 互いに秘密情報を持ち, かつ互いの秘匿情報を参照しなければならないという性質を有する麻雀将棋と呼ばれるボードゲームを採用する. 麻雀将棋とは, 最終的に相手の特定の牌(東)を取れば勝ちのゲームである. まず 25 個の麻雀牌の表面を自分の方向に向けて自由に設置し, 将棋盤上で牌を動かす. 互いの牌が衝突したときには予め定めている牌の強弱関係から第三者の審判が勝敗を判定し, 負けた牌が盤上から取り除かれる(引き分けの際は両方の牌が取り除かれる). このゲームを P2P 方式で設計するためには, (1)相手に自分の牌の初期配置を知られない一方, その配置をゲーム中にすり替えることができない, (2)牌の種類を互いに秘匿したまま, 勝敗結果のみ両プレイヤー間で共有する, といった計算を実行することが必要である. 本稿では, 公開鍵暗号技術を用いることで, これらのプロトコルを実現し, ネットワーク上でのサーバーレス麻雀将棋を設計し, 実際に実装した.

次節で麻雀将棋の説明を行い, 3 節で用いたプロトコルの説明と, そのプロトコルの妥当性と安全性について議論する. そして 4 節で実装においての具体的な案を述べた後, 5 節でまとめと今後の課題を述べる.

2. 麻雀将棋

麻雀将棋とは 1 人 25 個の麻雀牌(以下**コマ**と呼ぶ)と 9×9 マスの盤(将棋盤)を用いる二人用のボードゲームである. ここで, 手前から縦 1〜3 マス×横 9 マスの部分を**自陣**, 手前から縦 7〜9 マス×横 9 マスの部分を**相手陣**と呼ぶ. 用いるコマと盤を以下の図 1 に示す.

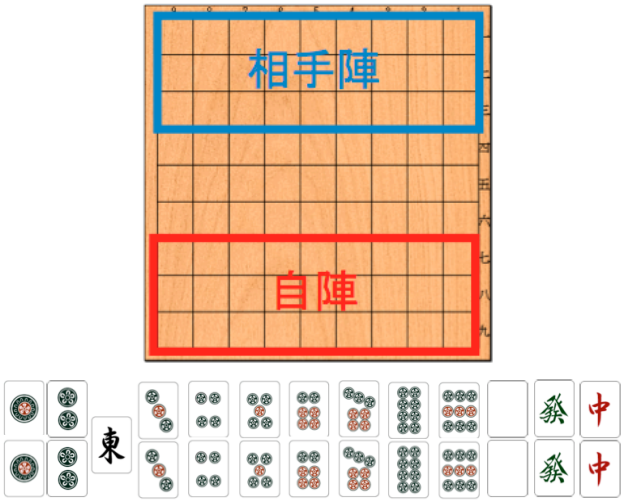

図1 : 麻雀将棋で用いるコマと盤

コマの強さは図1においては右にいくほど強い. ただし, 「一筒」(図1の一番左に記されているコマ)のみ「中」(図1の一番右に記されているコマ)に勝利する.

ゲーム開始時に, 各プレイヤーA, Bは自分の全てのコマを相手には見られない形で自陣に自由に配置する. このとき, 自分は相手のコマの配置を知ることができない.

プレイヤーが動かすことのできるコマの方向は上下左右の1マス分のみである. また, 互いのプレイヤーのコマが同じマス目に入ったとき, 前述したコマの強さに従い, 勝敗判定を行う. この勝敗判定は審判によって行われ, 勝利したコマを残し, 負けたコマを外に出す. 引き分けの場合は両方のコマを外に出す. コマを外に出す際, 審判はそのコマが何であるかをプレイヤーに見せないようにする.

ゲームは互いのプレイヤーが交互にコマを進めることで進めていき, 最終的には, 「東」(図1の左から三番目のコマ)を取った側のプレイヤーが勝利となる.

3. ネットワーク麻雀将棋プロトコルの設計

麻雀将棋を実装するにあたって, 審判サーバーを設置することができれば, 互いのコマの情報から審判が勝敗判定の結果のみを返すことが可能であるため, 各プレイヤーが相手のコマの情報を知るという不正はできない.

†京都大学 大学院情報学研究科, Graduate School of Informatics, Kyoto University
‡ 京都大学 学術情報メディアセンター, Academic Center for Computing and Media Studies, Kyoto University

一方で，審判サーバーを設置しないP2P方式のネットワーク上でゲームを設計するにあたっては，マルチパーティプロトコルによってゲームが進行する上での不正を検知する技術が必要である．

考えられる不正は，自分の初期配置の情報を相手が知るという不正と，勝敗判定を行う際，互いがその判定結果を共有するため，どちらが実際の判定とは異なる結果を不正に共有するという二つの状況である． これらの状況に対して不正を検出できるプロトコルを示す．

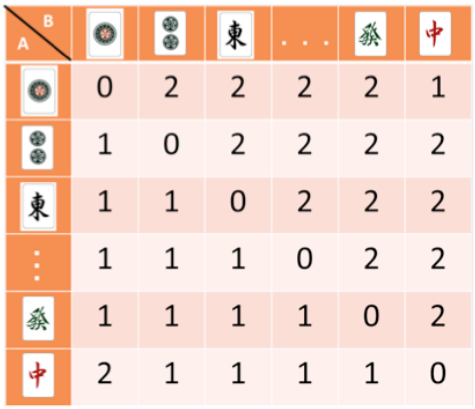

図2：勝敗判定関数

3.1 準備

プレイヤーはAとBとする． A，Bそれぞれの暗号化関数を$E_A(x)$, $E_B(x)$, 復号関数を$D_A(x)$, $D_B(x)$とおく． このとき，$D_A(E_A(x))=x$となる．また，$E_A(x)$, $E_B(x)$は**可換性**，つまり$E_A(E_B(x))= E_B(E_A(x))$を満たす公開鍵暗号方式を用い，$E_{AB}(x)= E_A(E_B(x))$とおく．可換性を満たす暗号系の具体例[4]を示す．

可換な暗号系

以下の1～3の操作をA，Bが行う．

1. 十分に大きな整数nをA，B間で共有する．
2. 整数nより小さく，かつオイラー関数φ(n)と互いに素な整数を一つ選択し，これをKとおく．（オイラー関数φ(n)とは，n以下でnと互いに素な自然数の個数.)
3. $L=K^{-1} \bmod \varphi(n)$ となるLを計算し，これを復号鍵とする．

また，xに対する一方向性ハッシュ関数を$H(x)$とおく．本稿でハッシュ関数はRSA暗号を用いて実装した．またA, Bのコマはそれぞれ$a_1,...,a_{25}$と$b_1,...,b_{25}$とおき，これらのコマがそれぞれ図1で示した13種類のいずれかのコマの種類を取る．このコマの種類の対応はランダムに決まるため，相手のコマ情報からコマの種類の情報を決定することはできない．勝敗判定関数は，k行k列のテーブルをtで表す．実際に用いるテーブルの例を図2に示す．

コマa_iとb_jが同じマス目に入ったときの，勝敗判定を$t(a_i,b_j)$と表す．$t(a_i,b_j)$はa_iが勝利すれば1, b_jが勝利すれば2, 引き分けなら0を返す．

3.2 不正検出可能なコマ配置のプロトコル

Aがコマa_iを種類xとして設置する手順．

1. Aはコマの種類xの下位4bit以外に乱数r_i を付与した値x'のハッシュ値$H(x')$とa_iをBに送信．
2. ゲーム終了時にAはa_iとx'をBに送信．

この手順を不正なく実行できたとき，Aはコマa_iを種類xとして設置することが可能であり，同様の手順をBに対しても適用できる．また，この手順は繰り返し適用することができる．

次に，このプロトコルによって排除できる不正と，その理由について述べる．

自分のコマが推定されることに対しての安全性

この手順でBの得る情報はa_iと$H(x')$で，ハッシュ関数の一方向性よりx'を推定することは難しい．ゆえに，xの推定はハッシュ関数の一方向性に依存して安全である．

相手の不当な操作に対しての安全性

Bが$H(x')$を記憶しておくことで，ゲーム終了時にAから送信される情報と照合することで不当な操作を検出することが可能である．

3.3 不正検出可能な勝敗判定のプロトコル

ここではAのコマa_iがBのコマb_jのいるマスに入って勝敗判定を行う，という状況を考える．

1. Aの操作（1）
   - Aはコマa_iの種類に応じた勝敗判定関数のテーブルの行の全ての要素の下位2bit以外を乱数$r_1,...,r_{13}$で埋める．
   - 得られた数列中の要素それぞれをE_Aで暗号化してBに送信．
2. Bの操作（1）
   - BはAから暗号化された数列：$E_A(t(a_i, 1)+r_1),...,E_A(t(a_i, 13)+r_{13})$を受信する．
   - 受信した数列からコマb_jの種類の番目の要素：$E_A(t(a_i, b_j)+r_j)$を選択する．
   - 選択した数列をE_Bで暗号化した値：$E_B(E_A(t(a_i, b_j)+r_1))$をAに送信する．
3. Aの操作（2）
   - Bから$E_B(E_A(t(a_i, b_j)+r_1))$を得る．
   - Aの復号関数D_Aを用いて復号することで，可換な暗号系であることより，$E_B(t(a_i, b_j)+r_j)$が得られ，これをBに送信する．
4. Bの操作（2）
   - Aから$E_B(t(a_i, b_j)+r_1)$を得る．
   - Bの復号関数D_Bを用いて復号することで，$t(a_i, b_j)+r_1$を得る．
   - この値の下位2bit以外を取り除いた値$t(a_i, b_j)$をAとBの間で共有する．

次に，このプロトコルによって排除できる不正と，その理由について述べる．

自分のコマが推定されることに対しての安全性

1. Aが Bから得られる情報は，$E_B(t(a_i, b_j)+r_j)$と$t(a_i, b_j)$である．$E_B(t(a_i, b_j)+r_j)$は$E_B(x)$の暗号方式の強度に依存して安全であるため，$t(a_i, b_j)+r_j$を推定することは困難である．また，$t(a_i, b_j)$とAの元々持っている情報a_iからb_jを推定することは，ゲームの性質上推定できる場合を除いて困難である．(ここでのゲームの性質上推定できる場合とは，例えば「引き分け」という

結果が返された場合, b_jがa_iと等しいと推定できる, という場合である.)

2. BがAから得られる情報は, $E_A(t(a_i, 1)+r_1),..., E_A(t(a_i, 13)+r_{13})$であり, これらが全て異なる値を持つということから, これらの情報から, $t(a_i, 1)+r_1,..., t(a_i, 13)+r_{13}$を推定することは$E_A(x)$の暗号強度に依存して困難である.

相手が不当な操作を行った場合の検出

1. Aの操作（１）において, Aが$E_A(t(a_i, 1)+r_1),..., E_A(t(a_i, 13)+r_{13})$を任意の値として送信する不正の検出.
   →Bがこれらの値を保存しておき, ゲーム終了時にAがD_Aとa_iを公開することで検証可能.
2. Aの操作（２）において, Aが$E_B(t(a_i, b_j)+r_j)$を任意の値として送信する不正の検出.
   →1と同様に検証可能.
3. Bの操作（１）において, Bが$E_B(E_A(t(a_i, b_j)+r_1))$を任意の値として送信する不正の検出.
   →Aがこの値を保存しておき, ゲーム終了時にBがD_Bとb_jを公開することで検証可能.
4. Bの操作（２）において, Bが$t(a_i, b_j)$を任意の勝敗結果としてAと共有する不正の検出.
   →3と同様にして検証可能.

また, これら二つの不正の検出は繰り返し適用でき, BがAのいるマス目に入った場合も同様である.

4. ネットワーク麻雀将棋の実装

実装は全てJava言語で行った.

初期配置は3.2節で説明したプロトコルに従って配置するため, データ構造としてA, Bのコマの位置情報を格納するデータ構造と, 相手コマの種類情報のハッシュを保存しておくデータ構造が必要となる. 実装にあたっては, 互いがコマを配置し, 接続要求を互いに行うことが確認できた時点で接続を行うという設計にした. 実際の初期状態の各プレイヤーの配置画面を以下の図3に示す.

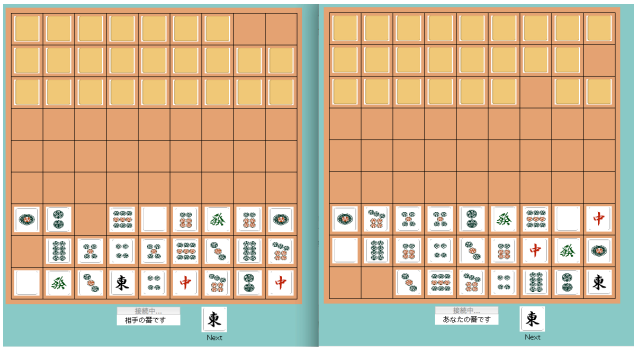

図3 : プレイヤーA, Bの初期状態の配置画面

勝敗判定が行われずに自分(または相手)のコマを動かす, という場合は, 自分のコマの位置情報を更新し, 相手に新たなコマの位置情報を送信し, 相手も同様にそのコマの位置情報の更新を行えばよい.

勝敗判定が行われる場合は, 3.3節のプロトコルに従った実装を行う. このとき, 互いに相手から送られてきた情報を順に記憶し, それぞれの情報が相手のどのコマにあたるか, ということも記憶するデータ構造(data log)が必要である.

ゲームが進行して勝敗が行われている画面を以下の図4に示す. (炎のエフェクトは, そのマスで対戦が行われていることを表す.)

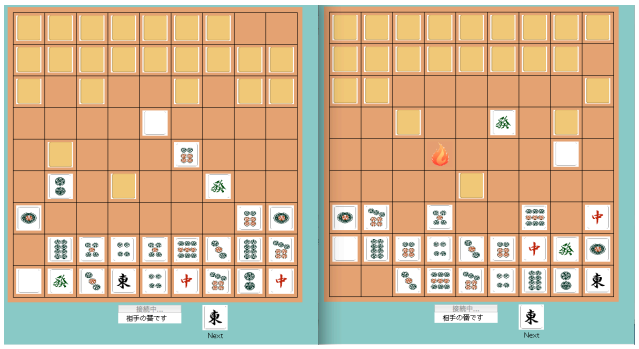

図4 : 対戦過程の画面

最終的な勝敗は, 「東」を取ったか取られたかで決定するため, その状況に応じて自分と相手に勝敗結果を表示する. そして, 勝敗が決まった後に, 相手から送信されたコマの種類情報を記憶し, logの順にコマの移動を実行することで, 勝敗結果とコマの移動可能性を検証していく. この検証によって不正が発覚した場合, 不正が行われたプレイヤーを通知し, 検証を中断する.

また, 検証が正しく行われたことを示すために, logを第三者に公開するという方法も考えられる.

5. まとめと今後の課題

本稿では, サーバーを設置することなく, 不正が行われないことを保証する麻雀将棋のプロトコルを示し, 実装した. P2P方式で行えるという手軽さから, ネットワーク対戦ゲームがこれから一層普及していくことが期待される. また, 本手法では, 互いが最終的に自分のコマを相手に公開することで不正の検証を行っているため, 何度かゲームを行う上で, 相手の初期配置の癖などが分かってしまう可能性がある. ゆえに, 今後の課題としては, 初期配置を相手に公開することなく, 不正の検出が可能なプロトコルを提案することが挙げられる.

参考文献

[1] 髪川, 千綿, 鷲見 : SIONet を適用した P2P 型ネットゲームシステム, *情報処理学会研究報告 GN,* pp. 39-44 (2002)

[2] W. Zhao, V. Varadharajan and Y. Mu : Fair on-line gambling, *Annual Computer Security Applications Conference,* pp. 394-400 (2000)

[3] 加藤, 宮崎, 岡部: 不正を検出できるネットワーク軍人将棋, *電子情報通信学会 信学技法,* Vol.103, No.499, pp. 1-6 (2003)

[4] A.Shamir, R. L. Rivest and L. M. Adleman : Mental Poker, *The Mathematical Gardner,* pp. 37-43 (1981)