**修士論文**

# P2P ネットワーク上でのオンラインゲームの安全な実行

指導教官　　岡部 寿男 教授

京都大学大学院情報学研究科
修士課程知能情報学専攻

西村 祐介

平成 17 年 2 月 9 日

# P2P ネットワーク上でのオンラインゲームの安全な実行

西村 祐介

**内容梗概**

現在のオンラインゲームは，サーバを用いる方式が主流である．この方式は，サーバを用いることでゲームの管理が容易になるという利点があるが，一方でサーバによる不正は防ぐことが不可能である．よって，サーバを利用しないピアトゥピア（P2P) 型のネットワークでオンラインゲームを安全に実行することを考える必要がある．

P2P 方式のオンラインゲームについては，ポーカーや軍人将棋など，個々のゲームに特化したプロトコルの提案や議論はなされている．しかしながら，一般のゲームを P2P 方式で実装する際に生じる問題点についての議論や，起こりうる不正についての考察が，厳密になされているものは見られない．

そこで本研究ではこれらの問題について議論をするため，ゲームを定式化し，そのモデルに基づいて，P2P 方式のオンラインゲームの安全な実行について考察を行った．

我々はまず順序機械を用いてゲームの定式化を行った．具体的には，ゲームの情報は機械の状態とし，各プレイヤのアクションは機械への入力，各プレイヤが見る情報は機械の出力とした．また，既存のゲームをこのモデルにより記述し，分類した．

次に，この定式化に基づいて，オンラインゲームを P2P ネットワークで実装する際の問題点を考察した．まず問題となる状態の管理については，秘密分散を利用した．また，秘密分散によって，全員にとって秘密の乱数を生成する手法を示した．さらに，マルチパーティプロトコルを利用することで，状態の秘密分散を維持したまま，状態遷移関数と出力関数を計算出来ることを示した．そして，この実装において考えられる不正を洗い出し，またその防止の可能性について議論した．その結果，与えられたゲームが，我々の定式化においてある種の条件を満たせば，それを P2P ネットワーク上で安全に実行できる方式が導かれることが言えた．

# Playing Online Games Securely on P2P Networks

Yusuke Nishimura

**Abstract**

When we play online games, one natural implementation is to use servers, which may easily control execution of games. However, the server itself may cheat, and in that case, it is impossible to prevent cheats by the server if our implementation completely relies on servers. This is the main motivation of research in this paper. We consider possibility of playing online games without servers, namely, using peer-to-peer (P2P) networks.

There have been a few work on secure P2P protocols which are customized for special games, such as poker and Gunjin-Shogi. Nevertheless, there seems to be no research on considering possibility of playing general online games on P2P networks.

In this thesis, we consider a problem of playing online games securely on P2P networks. We first formalize games using sequential machines. We regard game information as states, actions of each player as inputs, and information which each player can see as outputs. We also simulate existing popular games in our model to show that our model is sufficiently general, and classify those games by the degree of secretness of information and necessity of randomness.

We then consider problems of implementing online games on P2P networks. We show that the following four types of computation is essential: (i) putting a secret input into a sequential machine, (ii) generating a random bit that cannot be seen from players, (iii) computing a next state without knowing the current state and other player's inputs, and (iv) computing outputs without knowing the current state. We show that a slight modification of multi-party protocol resolves above problems. We finally discuss possibility of preventing cheats, and give some conditions that the game must satisfy to be played securely on P2P networks.

# P2P ネットワーク上でのオンラインゲームの安全な実行

# 目次

# 第1章　はじめに

インターネットの普及により，対戦型ゲームをネットワークを介して楽しむ，いわゆるオンラインゲームが盛んに行われるようになった [1]．対戦相手を見つけ，対面しながらゲームを行わなくてはならなかったゲームも，ネットワークを利用することにより，物理的に遠く離れた相手との対戦も可能であり，また自分の好きな時間に手軽に楽しむことができるという利点がある．

ゲームは予め定まったルールに従って進行していくが，参加者が少しでもルールに従わない行動（不正）をすればゲームを正常に進行できなくなる．さらにオンラインゲームは，相手が物理的に見えないという性質から，何らかの防御措置をとっていない限り容易に不正が行われる恐れがある [2][3]．

リアルタイムゲームでは，プレイヤの反応速度が重要な要素となり，公正さのため各プレイヤのアプリケーションでネットワークの遅延を考慮に入れた反応時間を申告することがあるが [4]，方式に従わない不正アプリケーションによって悪用される恐れがある．また，ネットワークの遅延をプレイヤに気づかせないようにするために，遅延によってプレイヤの行動が相手プレイヤに届かない場合に以前の行動から予測して表示させる方法があるが，この方法を悪用して自分の行動を知らせずに相手の行動のみ知り続ける不正の恐れがある．そこで，必要に応じてゲーム進行をロックして全員の情報が集まるまで待つ手法が考案されている [5]．このように，ネットワークの遅延からくる不正の問題に対して様々な研究がなされているが，ネットワークゲームではたとえ遅延が存在しない環境であっても起こりうる不正が存在する．

これまでのネットワークゲームで良く見られる形態は，ゲーム用のサーバを用意し，参加者がクライアントとなってサーバに接続することで，ゲームを進行していく，いわゆるクライアント／サーバ型のゲームである [6][7]．この方法の最大の利点は，サーバがゲームを完全に制御できるため，管理が簡単であると言うことである．先に挙げた不正の問題も，サーバが監視をすれば防ぐことが出来る．また，軍人将棋に代表されるような，第三者的な審判が必要であるゲームでは，それをサーバに行わせることが出来るため，クライアント／サーバ型のゲーム方式は適している．

しかし一方で，クライアント／サーバ型ではサーバが完全に信頼できなければならない．なぜなら，サーバはゲームの全ての権限を持っているため，サー

バはいくらでも不正をすることができ，これを防ぐことは不可能だからである．よって，全参加者が信頼できるような第三者機関によってサーバを運用する必要がある．しかしながらこのような機関は数が限られているので，そこに負荷が集中してしまう．また運用には多くのコストがかかる．このような背景から，サーバを用いないピアトゥピア型ネットワークゲでオンラインゲームを実現する手法が必要である．

ピアトゥピア方式の第一の問題点は，上述したようなゲームにおける審判の機能を，サーバなしでどのように実現するかである．例えば軍人将棋では，駒を伏せて相手に駒の種類が分からないまま動かし，駒がぶつかったときに第三者である審判が二つの駒の種類を見て，強い方を残すという動作が行われる．もちろん当事者同士は駒の種類を見てはいけない．しかも一般には，審判機能のみでなく，一部の参加者しか知り得ない情報からゲーム状態を次々に変化させていく機能が必要である．サーバを利用できる場合にはサーバが全ての情報を見れば良いので簡単に解決するが，ピアトゥピア型では容易ではない．

第二の問題点は，不正をどのように排除するかである．例えば将棋では，全ての情報が両対局者にオープンになっているため，不正を行うことができないが，ポーカーでは，相手の手札や場に積まれたカードの山は見ることが出来ない．それを利用し，例えば自分に有利なカードを持っているかのようにゲームを進め，高い役を作ったとしても，誰にもばれない可能性がある．また，例えば軍人将棋では，駒の種類を相手に知らせずにゲームを行うため，駒の配置の仕方がどのようになっているかを相手がチェックすることが出来ない．従って，駒の種類の制約に反して，例えば自分の駒全てを最強の種類のものとして初期配置を行い，その後通常にゲームが行われれば，圧倒的に勝てることになる．

二人用のポーカーでは，シャッフルの結果を都合よく操作したり山のカードを覗き見するなど，主に秘密の情報を生成することに対する不正が存在するが，これに対して公開鍵暗号技術を利用したプロトコルによって解決されている [8]．また軍人将棋についても解決策が提案されている [9]．しかしいずれも特定のゲームに特化した議論であり，一般のゲームに対する問題点を考察したものではない．

本研究では，これらの問題点を考察するためにゲームを定式化し，そのモデルに基づいて，ピアトゥピア型のオンラインゲームの安全な実行について議論する．

まずはじめに，ゲームを形式的に定義する．具体的には順序機械を用いて，各プレイヤのアクションを機械への入力とみなし，ゲームの情報は機械の状態，各プレイヤが得る情報は機械からの出力とする．また，従来のゲームがこのモデルにより記述可能であることを述べ，このモデルの一般性の高さを示す．さらに定式化に基づいてゲームを分類し，またゲームにおける不正についても考察する．

次に，この定式化にもとづいて，ゲームを実現する際の問題点について考察する．まずゲーム状態の管理が問題となるが，これは各プレイヤが状態を秘密分散して保持することで実現できるを述べる．また，秘密分散を利用することで，全プレイヤにとって秘密の乱数を生成する手法を述べる．次に，マルチパーティプロトコルを利用することで，状態の秘密分散を維持したまま，状態遷移関数と出力関数を計算出来ることを示す．そして，モデルのレベルで考察した不正に基づいて，実装レベルで起こりうる不正について考察する．

本論文の構成は次の通りである．まず第 2 章で関連研究としてマルチパーティプロトコルについて説明する．第 3 章ではゲームを順序機械により定式化した上で既存のゲームを記述・分類し，また不正についても考察する．第 4 章では，マルチパーティプロトコルを用いて，ピアトゥピア型のネットワーク上でゲームを実装する手法について述べ，またこの実装において起こりうる不正についても考察する．第 5 章では結論を述べる．

# 第2章　関連研究

本研究に必要な技術であるマルチパーティプロトコルと，それに関連する技術について説明する．

## 2.1　マルチパーティプロトコルとは

今，$n$ 人の参加者より構成されるネットワークがあるとして，各参加者 $P_i$ は，自分だけが知っている秘密の情報 $x_i$ を保持しているとする．ある関数 $f$ が与えられた時，各参加者は，各自の秘密情報を秘密にしたまま $y = f(x_1,\ \cdots,\ x_n)$ を計算しようとする．各参加者は，ネットワーク上で，各自の秘密を一切漏らさないで，お互いに何らかの通信を行い，最終的に全員が $y$ の値を正しく知ることが出来るような機能を実現する問題を考える．このような機能を実現するプロトコルを，関数 $f$ に対するマルチパーティプロトコルという．

## 2.2　1-out-of 2 紛失通信

マルチパーティプロトコルを実現する際に必要なのが，1-out-of-2 紛失通信である．送信者である Alice が，２つの情報 $m_0$ と $m_1$ を保持しており，そのうちのいずれか１つのみが受信者の Bob に伝わるが，Bob がどちらを受け取ったかを送信者の Alice は知ることが出来ない．1-out-of-2 紛失通信の実現方法として，公開鍵暗号方式によるものがある [10][11]．以下ではこれを紹介する．

1. Alice の公開鍵を $P_A$，対応する秘密鍵を $S_A$，また暗号化関数と復号関数をそれぞれ $E_{P_A},\ D_{S_A}$ とする．$D_{S_A}(E_{P_A}(x)) = x$ であり，また，$E_{P_A} : \mathbb{Z}_n \longrightarrow \mathbb{Z}_N$ とする．送信者である Alice は，ランダムな値 $r_0,\ r_1 \in_U \mathbb{Z}_N$ を選び，自分の公開鍵 $P_A$ と $r_0,\ r_1$ を Bob に送る．
2. 受信者である Bob は，ランダムに $b \in_U \{0,\ 1\}$ と $x \in_U \mathbb{Z}_n$ を選ぶ．さらに Bob は，
$$q = E_{P_A}(x) + r_b \mod N$$
を計算し，$q$ を Alice に送る．

3. Alice は，$y_i = D_{S_A}(q - r_i \mathrm{mod}\ N)(i = 0,\ 1)$ を計算する．さらに Alice は，

$$\begin{cases} c_0 = m_0 + y_0 \bmod n \\ c_1 = m_1 + y_1 \bmod n \end{cases}$$

を計算し，$(c_0,\ c_1)$ を Bob に送る．

4. Bob は $m_b = c_b - x \bmod n$ を得る．

このプロトコルの正当性であるが，まず，明らかに Bob は $m_b$ を得ることが出来る．一方，$m_{b\oplus 1}$ を求めることは，

$$D_{S_A}(q - r_i \bmod N) = D_{S_A}(E_{P_A}(x) + r_b - r_{b\oplus 1} \bmod N)$$

を求めることと等価である．したがって，この公開鍵暗号が安全であれば，$m_{b\oplus 1}$ を求めることは困難である．

## 2.3 マルチパーティプロトコルの実現法

ここでは，$n$ 人の全参加者が不正をしないことを前提とし，公開鍵暗号を用いて，任意の関数 $f$ に対してマルチパーティプロトコルを構成できることを示す [10][12]．任意の関数 $f$ に対しては，$f$ を計算する論理回路が構成できるため，この回路での論理演算のマルチパーティプロトコルの実現法を考える．このような回路は，通常 AND ゲート，OR ゲート，NOT ゲートを組合わせて構成されるが，OR は AND と NOT で実現できるため，AND と NOT で構成されている回路を考えればよい．

まず各参加者 $i$ は，秘密情報 $x_i$ を以下の関係を満たす $n$ 個のランダムな情報 $x_{i,1}, \cdots,\ x_{i,n}$ に分割し，$x_{i,j}$ を他の参加者 $j(j = 1, \cdots,\ n)$ に分配する．

$$x_i = x_{i,1} \oplus \cdots \oplus x_{i,n}$$

これ以降の計算は，全ての秘密が $n$ 人の間で分割されたままで行われる．したがって，いかなる計算中段階の情報も分割されたままであり，$n$ 人全員が協力しない限り情報を復元できない．最後に，$f$ の計算が終了した段階で，全員が各自の最後の計算結果を公開し，それら分散された計算結果情報を復元することにより，最終計算結果のみを全員が知ることが出来る．

情報を分割したまま，任意の回路の計算が出来ることを示すためには，AND

および NOT に対してそのような計算が出来ることを示せばよい．$a,\ b$ を AND ゲートへの入力，$c$ を AND ゲートの出力とし，$d$ を NOT ゲートへの入力，$e$ を NOT ゲートの出力とする．各参加者 $i$ は $a_i,\ b_i,\ d_i$ のみを知っているとする．

$$a = a_1 \oplus \cdots \oplus a_n,\ b = b_1 \oplus \cdots \oplus b_n,\ d = d_1 \oplus \cdots \oplus d_n$$

AND および NOT は，それぞれ $\bmod 2$ の掛け算および $+1$ に対応するため，各参加者 $i$ が $a_i,\ b_i,\ d_i$ のみを使って以下を満足するような $c_i,\ e_i$ を求められればよい．

$$c = ab \bmod 2,\quad e = d + 1 \bmod 2$$

$$c = c_1 \oplus \cdots \oplus c_n,\quad e = e_1 \oplus \cdots \oplus e_n$$

$e_i$ を求めるのは簡単である．例えば参加者 1 のみが $e_1 = d_1 + 1 \bmod 2$ として，それ以外の参加者は $e_i = d_i$ とすればよい．

次に，$c_i$ は以下のようにして計算する．

$$c_{ii} = a_i b_i,\ c_{ij} + c_{ji} \equiv a_i b_j + a_j b_i (\bmod 2)\ \ (i \neq j)$$

とすると，以下が成立する．

$$\begin{aligned} ab &\equiv (\sum_{i=1}^{n} a_i)(\sum_{i=1}^{n} b_i) \\ &\equiv \sum_{i=1}^{n} a_i b_i + \sum_{1 \leq i < j \leq n} (a_i b_j + a_j b_i) \\ &\equiv \sum_{i,\ j} c_{ij} \\ &\equiv \sum_{i=1}^{n} c_i \ (\bmod\ n) \end{aligned}$$

各参加者 $i$ は，$c_{ii}$ は自分で計算することが出来る．

$c_{ij},\ c_{ji}$ の計算方法については次の通りである．

1. 参加者 $i$ は，ランダムに $c_{ij} \in_U \{0,\ 1\}$ を選ぶ．
2. 参加者 $i$ は，

   $$s_{(\alpha,\ \beta)} = c_{ij} \oplus \beta a_i \oplus \alpha b_i\ (\alpha = 0,\ 1)\ (\beta = 0,\ 1)$$

   を計算し，$s_{(0,\ 0)}$，$s_{(0,\ 1)}$，$s_{(1,\ 0)}$，$s_{(1,\ 1)}$ を各参加者 $j$ に 1-out-of-4 紛失通信で送る．

3. 参加者 $j$ は，1-out-of-4 紛失通信で，４つの値，$s_{(0,\,0)}$，$s_{(0,\,1)}$，$s_{(1,\,0)}$，$s_{(1,\,1)}$ のうちの $s_{(a_j,\,b_j)}$ のみを手に入れる．そして，$c_{ji} = s_{(a_j,\,b_j)}$ とする．

上記のように計算された $c_{ij}, c_{ji}$ は確かに，

$$c_{ij} \oplus c_{ji} = c_{ij} \oplus (c_{ij} \oplus b_j a_i \oplus a_j b_i) = a_i b_j \oplus a_j b_i$$

となっている．

よって，参加者 $i$ は，$c_i = \sum_{j=1}^{n} c_{ij} \pmod 2$ を求めることができる．

# 第3章　ゲームモデル

本章ではゲームを定式化し，本研究で議論の対象となるゲームを考察する．

## 3.1　順序機械による定式化

本研究では次のようなゲームモデルを考える．ゲームの参加者（プレイヤ）の人数を $n$ とする．

**ゲーム**

- 入力集合： $I_1,\ I_2,\ \cdots,\ I_n$
- 出力集合： $O_1,\ O_2,\ \cdots,\ O_n$
- 状態集合： $S = \{s_0,\ s_1,\ \cdots,\ s_l\}$
- 初期状態：$s_0 \in S$
- 終了状態集合：$F \subseteq S$
- 乱数集合：$R$
- 状態遷移関数：$\delta : S \times I_1 \times I_2 \times \cdots \times I_n \times R \to S$
- 出力関数：$\lambda_1 : S \to O_1,\ \lambda_2 : S \to O_2,\ \cdots,\ \lambda_n : S \to O_n$

定式化には順序機械を用いた．ここでの順序機械は，$n$ 人のゲーム参加者（プレイヤ）$P_1,\ P_2,\ \cdots,\ P_n$ からの入力 $i_1,\ i_2,\ \cdots,\ i_n$ と乱数 $r$ の入力，そして各プレイヤへの出力 $o_1,\ o_2,\ \cdots,\ o_n$ を持つ．

入力集合 $I_k\ (k = 1, 2, \cdots, n)$ はそれぞれプレイヤ $P_k$ からの入力 $i_k$ の集合である．この入力 $i_k$ は，ゲームにおけるプレイヤ $P_k$ の行動に相当する．

出力集合 $o_k\ (k = 1, 2, \cdots, n)$ はそれぞれプレイヤ $P_k$ への出力 $o_k$ の集合である．この出力 $o_k$ は，ゲーム中にプレイヤ $P_k$ が見ることが出来る情報である．

状態集合 $S$ はゲームの状態 $s$ の集合である．$s_0 \in S$ は初期状態であるが，これはゲームにおいては何も意味しない，つまりビット列で考えるとすれば全て 0 の情報である．また，状態集合 $S$ は有限個の状態 $s_0,\ s_1,\ \cdots,\ s_l$ からなるとする．

終了状態集合 $F$ は，状態集合 $S$ の部分集合である．$F$ の各要素は終了状態である．つまり，ゲームの状態が $s \in F$ になった時，ゲームは終了する．

乱数集合 $R$ は文字通りゲームで使用する乱数 $r$ の集合である．多くのゲームでは乱数を使用するので，必要に応じて $r \in R$ を入力する．また，乱数を使用しないゲームでは，$R = \emptyset$ とする．

状態遷移関数 $\delta$ は，現在のゲームの状態 $s$ と，各プレイヤからの入力 $i$，そして乱数 $r$ を引数にとり，次の状態 $s'$ を計算する関数である．

出力関数 $\lambda_k$ $(k = 1, 2, \cdots, n)$ は状態 $s$ からプレイヤ $P_k$ への出力 $o_k$ を計算する関数である．

ゲームは次のように進行していく．各プレイヤはゲームにおける行動として，入力 $i$ を機械に与える．機械はそれらと乱数 $r$，現在の状態 $s$ および状態遷移関数 $\delta$ を用いて次の状態 $\delta(s, i_1, \cdots, i_n, r) = s'$ を計算し，ゲームの状態を $s'$ に遷移させる．次に機械は遷移後の状態 $s'$ と出力関数 $\lambda_k$ $(k = 1, 2, \cdots n)$ を用いて $\lambda_k(s') = o_k$ を計算し，計算結果 $o_k$ $(k = 1, 2, \cdots, n)$ をそれぞれプレイヤ $P_k$ に出力する．これを 1 ステップとする．このステップを繰り返すことでゲームを進めて行き，機械の状態を終了状態にすることをゲームの目的とする．

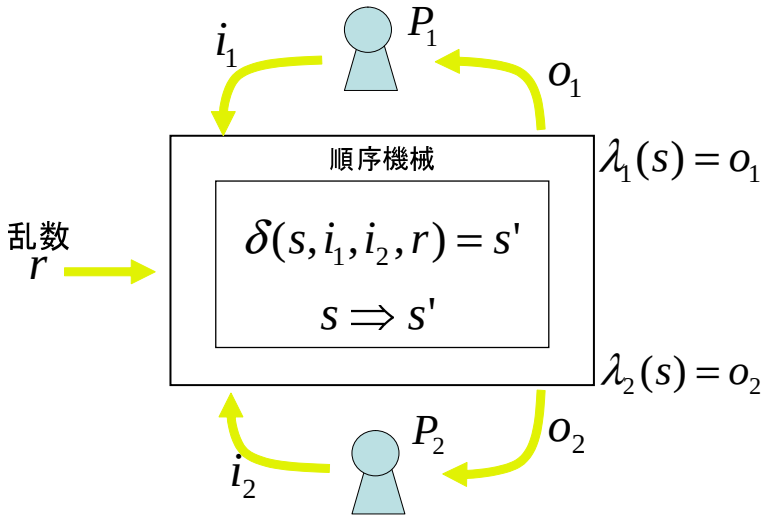


図 1: ゲームモデル

## 3.2 モデルによる具体的なゲームの記述

前述のモデルにより，具体的なゲームを記述する．これにより，このモデルの一般性の高さを示す．

### 3.2.1 将棋

まず，将棋を前述のモデルで記述する．

**将棋**

- $n = 2$(プレイヤを $P_1$ と $P_2$ とする)

- 入力集合 ： $I_1,\ I_2$
  ここで $I_k$ の各要素は指手に対応する．例えば，将棋の駒の位置は座標を用いて $(x,\ y)$ $(x = 0, 1, \cdots, 8)$ $(y = 0, 1, \cdots, 8)$ と表せるが，$(1, 2)$ にある駒を $(2, 3)$ に動かすことは，$((1, 2), (2, 3))$ を入力することに対応させればよい．

- 状態集合 ： $S$
  ここで $S$ の各要素は，$(a, b, t)$ の形をしている．$a$ は将棋の盤面の駒の配置を表し，$b$ は各プレイヤの持ち駒が何であるかを表す．
  $t$ は $0, 1, 2$ の値を取り，$t = 0$ は初期状態，$t = 1, 2$ はそれぞれ次が $P_1, P_2$ の手番であることを意味する．
  また，ゲーム開始状態は $s_1 = (a_1, b_1, 1)$ で，$a_1$ は初期盤面，$b_1$ は持ち駒なしという状態である．

- 初期状態： $(0, 0, 0)$

- 終了状態集合： $F = \{(a, b, t) | a$ は詰みの盤面 $\}$

- 出力集合： $O_1 = O_2 = S$

- 乱数集合： $\emptyset$

- 状態遷移関数： $\delta((a, b, t), i_1, i_2) = (a', b', t')$
  ただし、$\delta$ は以下の条件を満たす．
  - $t = 0$ のとき，
    $\delta((0, 0, 0), i_1, i_2) = (a_1, b_1, 1)$
  - $t = 1, 2$ のとき，$t' = 3 - t$ である．

また，$\delta$ は $i_{3-t}$ には依存しない．
形式的に書くと，例えば $t = 1$ のとき，任意の $i_1,\ i_2$ および $i_2'$ に対して

$$\delta((a, b, t), i_1, i_2) = \delta((a, b, t), i_1, i_2')$$

である．
$a'$ と $b'$ は $a$ と $b$ に入力 $i_t$ を加えたときに将棋のルールにのっとって得られる局面を表す．

- 出力関数：$\lambda_k(s) = s\ \ (k = 1,\ 2)$
  これは，プレイヤ $P_1,\ P_2$ がともに全状態を見る，ということを意味する．

### 3.2.2 ポーカー

ここで扱うポーカーのルールは次の通りである．なお，プレイヤは２人とする．

1. シャッフル：まずカードをシャッフルし，ランダムなカードの山を作る．
2. ディール：シャッフルされたカードの山から各プレイヤに５枚ずつカードが伏せたまま配られる．
3. チェンジ：プレイヤは自分が不要だと思ったカードを場に捨て，カードの山から捨てたのと同数のカードを再びとる．チェンジは一回のみとする．
4. オープン：プレイヤは全員が同時に自分の手持ちのカードを公開する．
5. 勝敗判定：あらかじめ決められた役のルールにより，勝敗を判定する．

このポーカーをモデルで表すと次の通りである．

**ポーカー**

- $n = 2$(プレイヤを $P_1$ と $P_2$ とする)

- 入力集合：$I_1, I_2$
  ここで $I_k$ の各要素は、プレイヤ $P_k$ が自分の手持ち札からチェンジフェーズで何を場に捨てるかを表す（52 枚のカードの部分集合族で，サイズ 5 以下のもの）．

- 状態集合：$S$

ここで $S$ の各要素は，$(a, c_1, c_2, t)$ の形をしている．
$a$ は山に積まれたカードの順列を表し，$c_k$ はプレイヤ $P_k$ の持ち札を表す．
$t$ は $0, 1, 2, 3$ の値を取り，$t = 0$ は初期状態を表し，$t = 1, 2$ はそれぞれ次が $P_1,\ P_2$ のチェンジフェーズであることを意味する．$t = 3$ はオープンフェーズ，$t = 4$ は全てが終わったところを意味する．
ゲーム開始状態は $s_1 = (a_1, c_{1_1}, c_{2_1}, 1)$ で，$a_1$ はシャッフルとディール後のカードの山の状態を表し，$c_{k_1}$ はプレイヤ $P_k$ の最初の持ち札を表す．

- 初期状態：$(0, 0, 0, 0)$

- 終了状態集合：$F = \{(a, c_1, c_2, 4) |\ \ a, c_1, c_2 : \text{可能な組合せ全て}\}$

- 出力集合：$O_k = \{(c_k, t)\} \cup \{(c_1, c_2, 4)\}\ \ (k = 1,\ 2)$

- 乱数集合：$\emptyset$

- 状態遷移関数：$\delta((a, c_1, c_2, t), i_1, i_2, r) = (a', c_1', c_2', t')$
  ただし、$\delta$ は以下の条件を満たす．
  - $t = 0$ のとき（シャッフル・ディールフェーズ），

    $$\delta((0, 0, 0, 0), i_1, i_2, r) = (a_1, c_{1_1}, c_{2_1}, 1)$$

    である．52 枚のカードの順列は 52! 通りであり，それぞれ 1 から 52! の番号を付する．$r$ は十分大きな乱数であり，$r \bmod 52!$ に対応する番号を付されたカードの順列を採用する．この並びのカードのうち，最初の 5 枚を $c_{1_1}$，次の 5 枚を $c_{2_1}$ とし，これら 10 枚を除いた残りのカードの順列を $a_1$ とする．
  - $t = 1$ のとき
    $\delta$ は $i_2$ に依存しない．形式的に書くと，任意の $i_1, i_2$ および $i_2'$ に対して，

    $$\delta((a, c_1, c_2, 1), i_1, i_2) = \delta((a, c_1, c_2, 1), i_1, i_2')$$

    である．

    $$\delta((a, c_1, c_2, 1), i_1, i_2) = (a', c_1', c_2', t')$$

とすると， $t' = 2,\ \ c'_2 = c_2$ であり，$a'$ は $a$ からプレイヤ $P_1$ が $i_1$ という操作で指定した枚数，山のうえからカードを取った状態である．

$c'_1$ は、$c_1$ から、プレイヤ $P_1$ が $i_1$ という操作で指定したカードを捨て，$a$ の上から当該枚数取ったものが加わった状態である．

- $t = 2$ のとき，$t = 1$ と同様である．
- $t = 3$ のとき，

  $\delta((a, c_1, c_2, 3), i_1, i_2) = (a', c'_1, c'_2, t')$

  とすると，$t' = 4,\ \ c'_k = c_k,\ \ a' = a$ である．

- 出力関数：$k = 1,\ 2$ に対し，

$$\lambda_k((a, c_1, c_2, t)) = \begin{cases} (c_k, t) & (t = 1 \text{ or } 2) \\ (c_1, c_2, 3) & (t = 3) \end{cases}$$

これは，$t = 1, 2$ のチェンジフェーズの時はお互いに相手の手札が見えず，$t = 3$ のオープンフェーズの時にはお互いの手札が見えることを表す．

### 3.2.3 軍人将棋

軍人将棋は図 2 に示すように，23 ～ 30× プレイヤ数 2 の駒と，横 6 ～ 9× 縦 8 ～ 9 の升目をもつ盤を用いた二人用ボードゲームである．

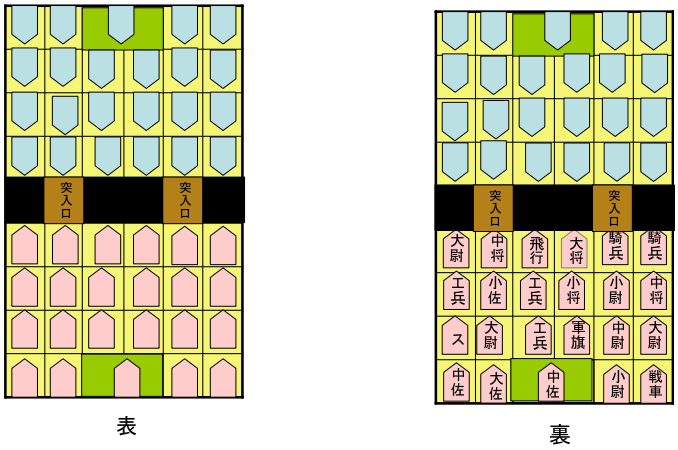


表　　裏

図 2: 軍人将棋

将官・佐官・尉官等の兵種とその階級によって命名された駒を用いる．盤は

自陣と敵陣に分かれ，各陣には司令部と呼ばれる枡がある．各プレイヤはゲーム開始時に自分の全ての駒を伏せて自陣に配置する．このとき，どの種類の駒をどの枡に配置したのかは相手には知らせない．

ゲームは，互いのプレイヤが交互に自分の駒を移動させることで進行する．駒は，その種類により一度に移動できる範囲が限定される．また，自分の駒が相手の駒と同じ枡に入ったとき，駒の種類によって勝敗を判定し，負けた駒を盤上から取り除く．引き分けの場合は両方の駒を取り除く．勝敗の判定は，図 3 のような勝敗判定表を用いる．

| | 大将 | 中将 | 少将 | 大佐 | 中佐 | 少佐 | ミサイル | 飛行機 | 地雷 | 砲兵 | 工兵 | タンク | スパイ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 大将 | --- | ○ | ○ | ○ | ○ | ○ | × | ○ | × | ○ | ○ | ○ | × |
| 中将 | × | --- | ○ | ○ | ○ | ○ | × | ○ | × | ○ | ○ | ○ | ○ |
| 少将 | × | × | --- | ○ | ○ | ○ | × | ○ | × | ○ | ○ | ○ | ○ |
| 大佐 | × | × | × | --- | ○ | ○ | × | × | × | ○ | ○ | × | ○ |
| 中佐 | × | × | × | × | --- | ○ | × | × | × | ○ | ○ | × | ○ |
| 少佐 | × | × | × | × | × | --- | × | × | × | ○ | ○ | × | ○ |
| ミサイル | ○ | ○ | ○ | ○ | ○ | ○ | --- | ○ | ○ | ○ | ○ | ○ | ○ |
| 飛行機 | × | × | × | ○ | ○ | ○ | × | --- | ○ | × | ○ | ○ | ○ |
| 地雷 | ○ | ○ | ○ | ○ | ○ | ○ | × | × | --- | × | × | ○ | ○ |
| 砲兵 | × | × | × | × | × | × | × | ○ | ○ | --- | ○ | ○ | ○ |
| 工兵 | × | × | × | × | × | × | × | × | ○ | × | --- | ○ | ○ |
| タンク | × | × | × | ○ | ○ | ○ | × | × | × | × | × | --- | ○ |
| スパイ | ○ | × | × | × | × | × | × | × | × | × | × | × | --- |

図 3: 勝敗判定表

この判定は通常審判が行い，互いの駒の種類が何であったのかを相手に知らせないよう配慮する．特定の駒を相手の司令部の枡に移動させ占領したプレイヤが勝者となる．

軍人将棋をモデルで表す場合，以下の準備をする．図 4 のように，駒は便宜上，表には一意となる識別番号が書かれており裏に駒の種類が書かれているものとする．各時点の盤面は，駒の識別番号とその盤上での位置を対応づけるこ

とにより表す．どの駒をどこに配置したかは，駒の識別番号と種類の対応により表すことができ，これが各プレイヤの秘密となる．

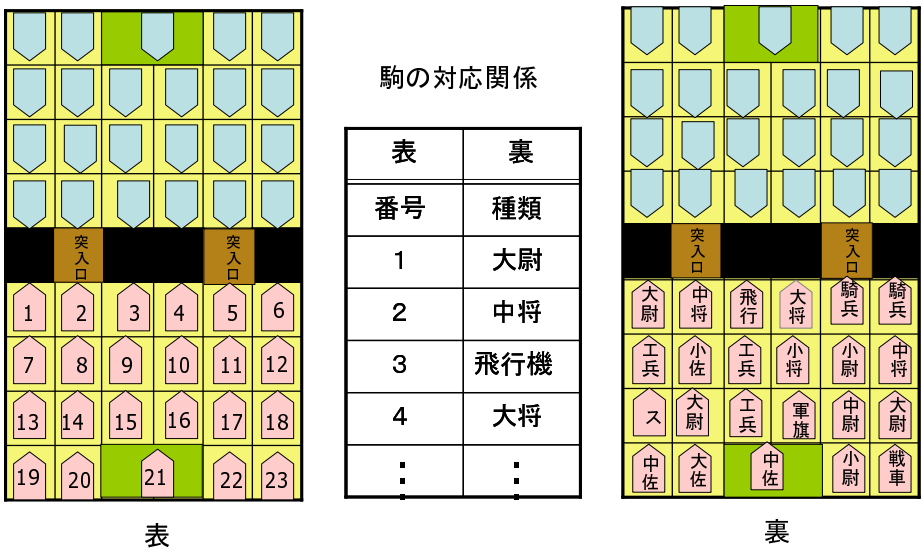


図 4: 駒の識別番号と種類の対応

モデルにより軍人将棋は次のように表せる．

**軍人将棋**

- $n = 2$(プレイヤを $P_1,\ P_2$ とする)

- 入力集合： $I_1, I_2$
  ここで $I_k$ の各要素はプレイヤ $P_k$ の指手に対応する．

- 状態集合：$S$
  ここで $S$ の各要素は，$(a, c_1, c_2, t)$ の形をしている．
  $a$ は将棋の盤面の駒の配置を表し，$c_k$ はプレイヤ $P_k$ の駒の識別番号と種類の対応関係を表す（$c_k$ はゲームの間中一定）．
  $t$ は $01, 02, 0, 1, 2$ の値を取り，$t = 0$ は初期状態，$t = 1, 2$ はそれぞれ次が $P_1, P_2$ の手番であることを意味する．また，$t = 0k$ の状態は，プレイヤ $P_k$ の指した後に駒がぶつかった状態（つまり判定してどちらかの駒を取り除く状態）である．

ゲーム開始状態は $s_1 = (a_1, c_{1_1}, c_{2_1}, 1)$ で，$a_1$ は初期盤面、$c_{k_1}$ はプレイヤ $P_k$ の駒の識別番号と種類の初期配置の対応関係を表す.

- 初期状態：$s_0 = (0, 0, 0, 0)$

- 終了状態集合：
  $F = \{(a, c_1, c_2, t) | a$ はどちらかが相手陣を占領した盤面 $\}$

- 出力集合：
  $O_1 = O_2 = \{(a, t) |$ 可能な組み合わせ全て $\}$

- 乱数集合：$\emptyset$

- 状態遷移関数：$\delta((a, c_1, c_2, t), i_1, i_2) = (a', c_1', c_2', t')$
  ただし、$\delta$ は以下の条件を満たす.
  - $t = 0$ のとき
    $$\delta((0, 0, 0, 0), i_1, i_2) = (a_1, c_{1_1}, c_{2_1}, 1)$$
    で，$c_{k_1} = i_k \ (k = 1,\ 2)$ である．つまり，ここではプレイヤ $k \ (k = 1,\ 2)$ は自分の駒の識別番号と初期配置の対応関係を入力することで，自身の駒の初期配置を行う.
  - $t = 1$ のとき
    $\delta$ は $i_2$ に依存しない．形式的に書くと，任意の $i_1, i_2$ および $i_2'$ に対して，
    $$\delta((a, c_1, c_2, 1), i_1, i_2) = \delta((a, c_1, c_2, 1), i_1, i_2')$$
    である.
    $$\delta((a, c_1, c_2, 1), i_1, i_2) = (a', c_1', c_2', t')$$
    とすると，$a'$ は、$a$ からプレイヤ $P_1$ が $i_1$ という操作で駒を動かした後の盤面である．$a'$ で両対局者の駒が同じ位置にいるときは $t' = 01$，そうでなければ $t' = 2$ である．また，$c_k' = c_k$ である.
  - $t = 2$ のとき

$\delta$ は $i_1$ に依存しない．形式的に書くと，任意の $i_1, i_2$ および $i_1'$ に対して，

$$\delta((a, c_1, c_2, 1), i_1, i_2) = \delta((a, c_1, c_2, 1), i_1', i_2)$$

である．

$$\delta((a, c_1, c_2, 1), i_1, i_2) = (a', c_1', c_2', t')$$

とすると，$c_k' = c_k$ である．

$a'$ は，$a$ からプレイヤ $P_2$ が $i_2$ という操作で駒を動かした後の盤面である．$a'$ で両対局者の駒が同じ位置にいるとき $t' = 02$，そうでなければ $t' = 1$ である．

- $t = 01$ のとき

  $\delta$ は $i_1, i_2$ に依存しない．形式的に書くと，任意の $i_1, i_2, i_1', i_2'$ に対して，

  $$\delta((a, c_1, c_2, 01), i_1, i_2) = \delta((a, c_1, c_2, 01), i_1', i_2')$$

  である．

  $$\delta((a, c_1, c_2, 01), i_1, i_2) = (a', c_1', c_2', t')$$

  とすると，$a'$ は，$a$ の盤面で駒がぶつかったところから，負けた方の駒を取り除いた盤面を表す．また，$c_k' = c_k \ \ (k = 1,\ 2)$ であり，$t' = 2$ である．

- $t = 02$ のときは，$t = 01$ の場合と同様である．

- 出力関数：$\lambda_1((a, c_1, c_2, t)) = \lambda_2((a, c_1, c_2, t)) = (a, t)$
  これはゲーム中，プレイヤ $P_1, P_2$ ともに盤面を見ることが出来るが，駒の初期配置を見ることが出来ないことを表す．

### 3.2.4 双六

ここでは単純な双六を扱う．参加者 2 人が交互にサイコロを振り，出た目の数だけマップ上の自分の駒を進めて行き，最初にゴールに到着した方を勝者とする．これをモデルにより記述する．

**双六**

- $n = 2$ (プレイヤを $P_1, P_2$ とする)

- 入力集合： $I_1, I_2$
  $I_k = \{0, 1, 2, \cdots, 5\}$ とし，この要素はサイコロを振るアクションを表す.

- 状態集合：$S$
  ここで $S$ の各要素は $(a, t)$ の形をしている.
  $a$ はマップ上の各駒の位置を表す.
  $t$ は $0, 1, 2$ の値をとり，$t = 0$ は初期状態，$t = 1, 2$ はそれぞれプレイヤ $P_1, P_2$ の手番であることを表す.
  ゲーム開始状態は $s_1 = (a_1, 1)$ で，$a_1$ はプレイヤの駒がスタート地点にある場面を現す.

- 初期状態：$s_0 = (0, 0)$

- 終了状態集合：$F = \{(a, t) | a$ はどちらかの駒がゴールに到着している場面 $\}$

- 乱数集合：$R$

- 出力集合：$O_k = \{(a, t) |$ 可能な組合せ全て $\}$ $(k = 1, 2)$

- 状態遷移関数：$\delta((a, t), i_1, i_2) = (a', t')$ ただし $\delta$ は以下の条件を満たす.
  - $t = 0$ のとき
    $\delta((0, 0), i_1, i_2, r) = (a_1, 1)$
  - $t = 1, 2$ のとき
    $\delta((a, t), i_1, i_2, r) = (a', t')$ とすると，$t' = 3 - t$ で，このとき $\delta$ は $i_{3-t}$ には依存しない．プレイヤ $P_t$ は入力 $i_t$ によりサイコロを振ったとみなされ，振って出た目の数は，$(r + i_t \bmod 6) + 1$ とする．$a'$ は，$a$ の場面からサイコロを振って出た目の数だけプレイヤ $P_t$ の駒を動かした後の場面を表す.

- 出力関数：$\lambda_k(s) = s \ \ (k = 1,\ 2)$
  これは，プレイヤ $P_1, P_2$ ともにゲームの全状態を見ることが出来ることを表す.

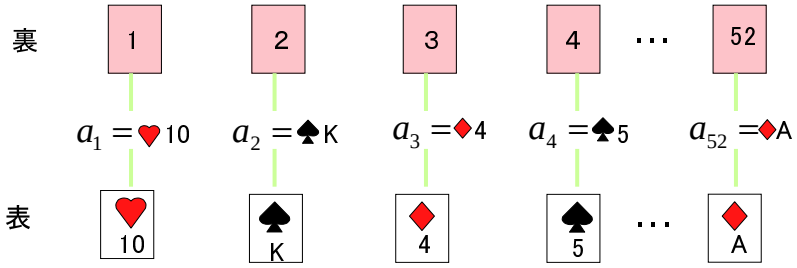


図 5: カードの識別番号と種類の対応

### 3.2.5 神経衰弱

ここでの神経衰弱も単純なものを扱う．最初は，52 枚のカードが伏せてランダムに並べられている．手番のプレイヤが 2 枚めくり，それらが同じ数字であれば自分のカードとしてその場から取り，続けてめくることができる．めくったカードが異なる数字であれば，手番は次の人に移る．最終的により多くのカードを取った者を勝者とする．さらにここではプレイヤの数を 2 人とし，2 枚のカードは同時にめくるものとする．

この神経衰弱をモデルで記述するため，次のような準備をする．使用する 52 枚のカードの裏面には，一意の識別番号 $1, 2, \cdots, 52$ がついている．また，カードの種類を $a_1, a_2, \cdots a_{52}$ とする．識別番号 $k$ のカードの種類が $a_k$ であるとする．これにより，伏せているカード $k$ を表にする行為は，$(k, a_k)$ を知ることと同一視することが出来る．

**神経衰弱**

- $n = 2$ （プレイヤを $P_1, P_2$ とする）

- 入力集合：$I_1, I_2$
  $I_k$ の各要素は，どのカードをめくるかを表す．
- 状態集合：$S$
  $S$ の各要素は $(a, b, c, p_1, p_2, t)$ の形をしている．
  $a = (a_1, a_2, \cdots, a_{52})$ で，$a_k$ は，識別番号 $k$ のカードの種類を表す．ゲーム中 $a$ は常に一定である．$b$ は場に残っているカードの識別番号を表す．
  $c$ は，表にされたカードを表す．例えば，識別番号 $j, k$ のカードが表にされた場合，$c = ((j, a_j), (k, a_k))$ とする．

$p_1, p_2$ はそれぞれプレイヤ $P_1, P_2$ のカードの獲得枚数を表す.
$t$ は $0, 1, 2, 01, 02$ の値をとり，$t = 0$ は初期状態，$t = 1, 2$ はそれぞれプレイヤ $P_1, P_2$ の手番ということを表す．$t = 01, 02$ はそれぞれプレイヤ $P_1, P_2$ が 2 枚のカードを表にした直後の状態を表す.
ゲーム開始状態は $s_1 = (a, b_1, 0, 0, 0, 1)$ で，$b_1 = (1, 2, 3, \cdots, 52)$ である.

- 初期状態：$s_0 = (0, 0, 0, 0, 0, 0)$

- 終了状態集合：$F = \{(a, b, c, p_1, p_2, t) | b$ は全てのカードがなくなった場面 $\}$

- 出力集合：$O_k = \{(b, c, p_1, p_2, t) |$ 可能な組合せ全て $\}$ $(k = 1, 2)$

- 乱数集合：$R$

- 状態遷移関数：
  $\delta((a, b, c, p_1, p_2, t), i_1, i_2, r) = (a, b', c', p_1', p_2', t')$
  ただし，$\delta$ は以下の条件を満たす.
  - $t = 0$ のとき
    $\delta((0, 0, 0, 0, 0, 0), i_1, i_2, r) = (a, b_1, 0, 0, 0, 1)$
    $b_1 = (1, 2, 3, \cdots, 52)$ である．最初の 52 枚のカードの配置方法は 52! 通りであり，それぞれの並び方に 1 から 52! の番号を付する．$r$ は十分大きな乱数であり，$r \bmod 52!$ に対応する番号を付されたカードの配置を $a$ とする.
  - $t = 1$ のとき
    $\delta$ は $c, p_1, p_2, i_2, r$ には依存しない．$b' = b,\ p_1' = p_1,\ p_2' = p_2$ であり，$i_1 = (m_1, m_2)$ とすると，$c' = ((m_1, a_{m_1}), (m_2, a_{m_2}))$ である．また，$t' = 01$ である.
  - $t = 2$ のとき
    $t = 1$ の場合と同じである.
  - $t = 01$ のとき
    $\delta$ は $a, i_1, i_2, r$ に依存しない．$c' = ((0, 0), (0, 0))$ である．$c = ((m_1, a_{m_1}), (m_2, a_{m_2}))$ とすると，$a_{m_1} = a_{m_2}$ ならば，$b$ の識別番号列から $m_1, m_2$ を除いた

ものを $b'$ とし，$p'_1 = p_1 + 2,\ t' = 1$ とする．$a_{m_1} \neq a_{m_2}$ ならば，$b' = b,\ p'_1 = p_1,\ t' = 2$ とする．

- $t = 02$ のとき
  $t = 01$ の場合と同じである．

- 出力関数：$\lambda_k(a, b, c, p_1, p_2, t) = (b, c, p_1, p_2, t)\ \ (k = 1,\ 2)$
  これはプレイヤ $P_1, P_2$ はともに，カードの識別番号と種類の対応関係以外のゲーム状態を見ることが出来ることを表す．

## 3.3 モデルで表せないゲーム

上述のように，多くのゲームがこのモデルにより記述出来る．しかしながら，一部のゲームの中にはこのモデルで記述できないものがある．それは次のようなものである．

- **実時間の要素が必要であるゲーム**
  このモデルのゲーム状態は離散時間で遷移する．よって，トランプのスピードのように，実時間の要素が必須のゲームはこのモデルでは記述出来ない．

- **ゲーム状態が無限に存在するもの**
  このモデルの状態集合は有限集合であるから，状態が無限に存在するゲームはこのモデルでは記述出来ない．例えば，無限の広さの盤上における五目並べなどがそうである．

以上のように，モデルにより記述出来ないゲームは本研究では議論の対象としないことにする．

## 3.4 ゲームにおける情報の分類

ゲームにおける情報を以下のように分類する．

- **公開情報**
  プレイヤ全員が知ることができる情報．

- **個人情報**
  あるプレイヤのみが知ることができる情報.
- **非公開情報**
  プレイヤ全員が知ることができない情報.

例えば，将棋におけるゲームの情報はすべて公開情報である．なぜなら，両プレイヤは盤上の駒を全て見ることが出来るからである．また，ポーカーにおいては，各プレイヤは自分の手札しか見ることが出来ず，他のプレイヤの手札と場にあるカードの山は見ることが出来ない．よって，各プレイヤの手札は個人情報であり，場に積まれたカードの山は非公開情報である．

## 3.5 ゲームの分類

### 3.5.1 3要素による分類

ここではゲームを以下の基準で分類する．

1. **ランダム要素の有無**
   - ランダム要素のあるもの：$R \neq \emptyset$
   - ランダム要素のないもの：$R = \emptyset$
2. **状態の秘匿性の有無**
   - 状態の秘匿性がないもの：
     $\forall k \in \{1, 2, \cdots, n\},\ \forall s \in S,\ \lambda_k(s) = s$
   - 状態の秘匿性があるもの：
     $\exists k \in \{1, 2, \cdots, n\},\ \exists s \in S,\ \lambda_k(s) \neq s$
3. **出力の秘匿性の有無**
   - 出力の秘匿性がないもの：
     $\lambda_1 = \lambda_2 = \cdots = \lambda_n$
   - 出力の秘匿性があるもの：
     $\exists j, k \in \{1, 2, \cdots, n\}\ (j \neq k),\ \lambda_j \neq \lambda_k$

上の分類においては，状態の秘匿性がないゲームは出力の秘匿性もない．よってゲームは表１のように６つのカテゴリーに分類できる．

表１のＢ以外のカテゴリーに属するゲームについては3.2節で述べたので，ここではカテゴリーＢに属する「HIT&BLOW」というゲームについて説明する．

表 1: ゲームの分類

| | 状態秘匿 | 出力秘匿 | ランダム性 | 主なゲーム |
|---|---|---|---|---|
| A | ○ | ○ | ○ | ポーカー |
| B | ○ | ○ | × | HIT&BLOW(プレイヤ数 3 以上) |
| C | ○ | × | ○ | 神経衰弱 |
| D | ○ | × | × | 軍人将棋 |
| E | × | × | ○ | 双六 |
| F | × | × | × | 将棋 |

このゲームはまず，各プレイヤがそれぞれ 0,1,2,3,4,5,6,7,8,9 の 10 個の数から異なる 4 つを任意に選び，その 4 個の数で 4 桁の数を作る．ゲームの目的は相手が作った 4 桁の数を当てることである．

先攻プレイヤは任意の 4 桁の数を言う．相手はその言われた数と自分が作った数について，位置と数字が一致していれば「HIT」，位置は違うが数が含まれていれば「BLOW」とし，HIT の数と BLOW の数を先攻プレイヤに伝える．例えば，プレイヤ $P_1$ が作った数字が「1234」だったとする．プレイヤ $P_2$ が $P_1$ に「1246」と尋ねたとき，プレイヤ $P_1$ は「2 HIT, 1 BLOW」と答える．このような質問と返答を交互に行い，互いに相手の数を予想してゆき，はじめに相手の数を当てた方が勝者となる．

但し，表 1 のカテゴリー B に含まれる HIT&BLOW は 3 人以上で行う場合で，以下のような特殊なケースである．各プレイヤが自分の数字を決めるところまでは同じである．次に，各プレイヤが順々に質問をする．この際，任意の相手を指名し，質問と返答を行うが，この 2 人の当事者以外はこのやりとりを一切知りえない．つまり，各々のプレイヤは自分の力だけで他のプレイヤの数字を当てなければならない．この条件の下，対戦プレイヤ全員の数を最初に当てることが出来たものを勝者とする．

### 3.5.2 カテゴリー間の関係

前述の分類では，各カテゴリーが互いに素とならない．例えば，先述のモデルによる軍人将棋の記述において，各プレイヤの初期配置情報 $c_k$ をそれぞれのプレイヤに毎ステップ出力することにする．このとき，出力関数 $\lambda_1,\ \lambda_2$ は $\lambda_1 \neq \lambda_2$

となり，出力秘匿ありのゲームとなるので，表1のカテゴリーにおいてはDに含まれるゲームとなる．

状態秘匿なし，あるいは出力秘匿なしのゲームは，各出力関数にそれぞれ異なる，適当な情報を加えることにより，あたかも状態秘匿がある，あるいは出力秘匿があるかのようなゲームになってしまう．つまり，カテゴリーの包含関係として，$C \subseteq A,\ D \subseteq B$ などが成り立つことになる．

## 3.6 ゲームにおける不正

ここでは先述のゲームモデルに基づいて，ゲームにおいて考えられる不正について考察する．

不正の種類としては大きく分けて以下の４つがある．

- **漏洩**
  自分が所有している秘密の情報を，他人に送る．
- **盗聴**
  見てはならない情報を不当に盗み見る．
- **改ざん**
  情報を不当に違う内容に改める．
- **否認**
  自分が行った行動について、後で否認する．

これを基本とし，本研究のゲームモデルにおいて起こりうる不正を以下に列挙する．

- **ゲームの存在の否定**
  ゲーム終了後，そのようなゲームはしていないと主張する．
- **ゲームの停止**
  入力 $i$ を与えない．これにより，状態遷移関数 $\delta$ を計算することが出来ず，ゲームの状態 $s$ を遷移させることが出来ない．
- **不当な入力**
  入力として $i \notin I$ を与える．あるいは，ルールに反する入力を行う（例えば先述の将棋において，歩を斜めに動かす行為）．

- **入力の漏洩**
  自分の入力 $i$ を他人に見せる．
- **入力の否認**
  入力 $i$ を与えた後，そのような入力をしていないと主張する．
- **入力の盗聴・改ざん**
  他人の入力 $i$ を盗み見る．また，他人の入力 $i$ を $i'(\neq i)$ にすりかえる．
- **乱数の盗聴・改ざん**
  乱数 $r$ を盗み見る．また，乱数 $r$ を，自分にとって都合の良い値 $r'(\neq r)$ にすりかえる．
- **出力の漏洩**
  自分の出力 $o$ を他人に見せる．
- **出力の盗聴・改ざん**
  他人の出力 $o$ を盗み見る．また，他人の出力 $o$ を $o'(\neq o)$ にすりかえる．
- **ゲーム状態の盗聴・改ざん**
  状態 $s$ を盗み見る．また，状態 $s$ を $s'(\neq s)$ にすりかえる．
- **状態遷移関数の改ざん**
  状態遷移関数 $\delta$ を $\delta'(\neq \delta)$ とすりかえる．
- **出力関数の改ざん**
  出力関数 $\lambda$ を $\lambda'(\neq \lambda)$ とすりかえる．
- **ゲーム情報の盗聴**
  任意の関数 $f$ に対し，$f(s)$ ($s$ : 状態 ) を計算し，結果を見る．
- **結託**
  数人のプレイヤが，お互いの秘密の情報を共有するなど，協力をしてゲームを進めること．

# 第4章　P2P方式でのゲームの実装

本章では上述のモデルを用いて，ピアトゥピア型ネットワークでオンラインゲームを実現する際の問題点について考察する．

サーバを利用したオンラインゲームであれば，正常なゲームの進行およびクライアントの不正の排除は容易である．なぜなら，状態 $s$ を管理し，各プレイヤからの入力を受け取り，次状態 $s$ を計算し，各プレイヤに出力 $o$ を送る，という一連の作業を全てサーバが担えばよいからである．不正な入力や状態の盗聴は，サーバにより排除できるのである．

一方，ピアトゥピア方式のオンラインゲームでは，不正の排除は勿論，ゲームをルール通り進行させることでさえも自明ではない．なぜなら，上述のゲームにおける一連の作業をサーバなしで，つまり参加者だけで行わなければならず，しかもその過程で例えば状態 $s$ や他プレイヤへの出力 $o$ などは見えてはならないからである．

本章ではまず不正者がいない場合，ゲームをルールに基づいて進行させる際に問題となる点を挙げ，それに対し秘密分散とマルチパーティプロトコルを利用した解決方法を示す．次に，不正者が存在する場合について，いかなる不正が排除できるかについて考察する．

## 4.1　ネットワークの前提

個々の議論の前に，ここで用いるピアトゥピア型のネットワークの前提を述べる．まず，各プレイヤは他の全プレイヤとの通信路を持つ．また，暗号化通信やディジタル署名の技術を利用することで，通信情報の盗聴・改ざん・なりすましは不可能であるとする．

また本研究では離散時間で状態が遷移するゲームを対象とし，そのようなゲームの定式化を行ったが，これによりネットワークの遅延を直接の考慮の対象から切り分ける．

## 4.2　不正者がいない場合のゲームの進行

### 4.2.1　ゲーム進行における問題

ここでは，どのプレイヤも不正者ではなく，全員が協力してゲームを進行させることとする．まず，問題となるのが状態 $s$ の管理である．ピアトゥピア型で

は，状態 $s$ はプレイヤだけで管理しなければならないが，例えば先のポーカーなどのようなゲームは，ゲーム情報の全体，つまり状態 $s$ が全て見えてしまうとゲームとして成立しない．よって，この状態 $s$ を参加者の誰からも見えない形で，参加者の間で管理しなければならない．

これを解決する手法として，秘密分散がある．秘密分散の最も単純な方法は次のものである [13]．状態 $s$ を，

$$s = s_1 \oplus s_2 \oplus \cdots \oplus s_n$$

と分割し，参加者 $P_i$ が $s_i$ を保持すればよい $(i = 1,\ 2,\ \cdots,\ n)$．この場合，$s$ の分割を行うのはプレイヤ以外の第三者でなければならないが，P2P 型のオンラインゲームではこのような第三者の存在を仮定しない．よってゲーム進行途中ではこのような分散は行うことは出来ない．しかし，初期状態は $s_0 = 0$ で各プレイヤにとって既知であるから，各プレイヤ $k$ が $s_{0_k}$ を保持することで

$$s_0 = s_{0_1} \oplus \cdots \oplus s_{0_n}$$

という分散を実現できる．この時，問題となるのが次の 4 点である．

- **入力の秘匿**
  各プレイヤの入力は，他のプレイヤに見えない形で保持しなければならず，また状態遷移関数の計算にも使えなければならない．
- **乱数の秘匿生成**
  先述のモデルにおける乱数の入力は，全プレイヤから見えてはならない．ここでは $n$ 人のプレイヤだけでこのような乱数を生成しなければならない．またその生成された乱数は，状態遷移関数の計算に使えなければならない．
- **状態遷移関数の計算**
  信頼できる第三者がいないので，プレイヤだけで状態遷移関数 $\delta$ を正しく計算しなければならない．しかも，秘密分散を維持しなければならないので，計算結果 $s'$ も分散された形で得られなければならない．
- **出力関数の計算**
  信頼できる第三者がいないので，プレイヤだけで出力関数 $\lambda$ を正しく計算し，そしてその $\lambda_k$ の計算結果 $o_k$ はプレイヤ $P_k$ にのみ見えるようにしなければならない．

以降，これらの問題点の解決法について述べる．

### 4.2.2 入力の秘匿

各プレイヤの入力を秘匿にするには，先述の秘密分散が利用できる．プレイヤ $P_k$ が入力 $i_k$ を与えるには次のようにすればよい．まずプレイヤ $P_k$ は入力 $i_k$ を次のように分割する．

$$i_k = i_{k_1} \oplus \cdots \oplus i_{k_n}$$

そしてプレイヤ $P_k$ は，分割された $i_{k_j}$ をプレイヤ $P_j$ に送る．

これが状態遷移関数の計算に利用可能であることは 4.2.4 で述べる．

### 4.2.3 乱数の秘匿生成

先述のモデルにおける乱数の入力は各プレイヤに対して秘匿であるが，そのような乱数入力をここでは $n$ 人のプレイヤだけで実現させなければならない．ここでも先述の秘密分散を利用する．

1 ビットの乱数 $r$ を生成し秘密分散するには，各プレイヤ $P_k$ がランダムに $r_k \in \{0, 1\}$ を選び，

$$r = r_1 \oplus r_2 \oplus \cdots \oplus r_n$$

とすればよい．実際 $r$ を計算せずに，$r$ を秘密分散させたものとして各プレイヤ $P_k$ が $r_k$ を保持すればよい．各プレイヤが故意に数字を選んだとしても，少なくとも一人のプレイヤ $P_k$ が $r_k$ をランダムに選べば，$r$ は一様にランダムな数となる．

この方法により，任意の $m$ に対し $m$ ビットの乱数 $r$ を秘匿生成することが出来る．このように生成された $r$ は，

$$r = r_1 \oplus r_2 \oplus \cdots \oplus r_n$$

と分散され，各プレイヤ $P_k$ が $r_k$ を保持しているが，これを状態遷移関数の計算において使用できることは次に述べる．

### 4.2.4 状態遷移関数の計算

ここでは，状態 $s$ の秘密分散を維持したまま，状態遷移関数 $\delta$ が計算できることを示す．

まず，遷移前の状態は秘密分散されていると仮定する．つまり，遷移前の状態 $s$ は，

$$s = s_1 \oplus \cdots \oplus s_n$$

と分散されていて，プレイヤ $P_k$ が $s_k$ を保持しているとする．

$\delta$ の計算には各プレイヤの入力と乱数を用いるが，入力については先述の通り，プレイヤ $P_k$ の入力 $i_k$ を

$$i_k = i_{k,1} \oplus \cdots \oplus i_{k,n}$$

と分割し，$i_{k,j}$ をプレイヤ $P_j$ $(j = 1, 2, \cdots, n)$ に送る．乱数についても先述の方法を用いて生成し，

$$r = r_1 \oplus \cdots \oplus r_n$$

なる $r_k$ をプレイヤ $P_k$ が保持する．

次の状態 $s'$ を計算するには，プレイヤ $P_k$ が $s_k$ を，

$$s_k = s_{k,1} \oplus \cdots \oplus s_{k,n}$$

と分割し，$s_{k,j}$ をプレイヤ $P_j$ $(j = 1,\ 2,\ \cdots,\ n)$ に送る．

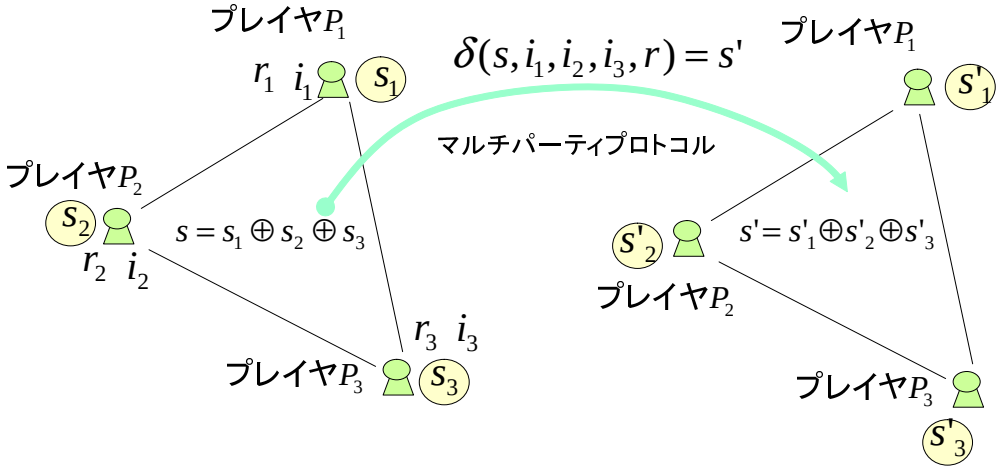


図 6: 状態遷移関数の計算

ここで，先述のマルチパーティプロトコルを状態遷移関数 $\delta$ に適用することで，プレイヤ $P_k$ は以下を満たす $s'_k$ を他の参加者に知られない形で得る．

$$\begin{cases} \delta(s,\ i_1,\ \cdots,\ i_n,\ r) = s' \\ s' = s'_1 \oplus \cdots \oplus s'_n \end{cases}$$

よって，秘密分散を維持した形で，状態遷移関数を計算出来ることが帰納的に示せた．

### 4.2.5 出力関数の計算

出力関数を計算するには次のようにすればよい．各参加者 $P_k$ は自分が保持している $s_k$ を

$$s_k = s_{k,1} \oplus \cdots \oplus s_{k,n}$$

と分割し，$s_{k,j}$ を参加者 $P_j$ $(j = 1,\ 2,\ \cdots,\ n)$ に送る．そして，前述のマルチパーティプロトコルを出力関数 $\lambda_m$ に適用することで，参加者 $P_k$ は以下を満たす $o_{m,k}$ を他の参加者に知られない形で得る．

$$\begin{cases} \lambda_m(s) = o_m \\ o_m = o_{m,1} \oplus \cdots \oplus o_{m,n} \end{cases}$$

そして，各参加者 $P_k$ は計算結果 $o_{m,k}$ を参加者 $P_m$ に送り，$P_m$ は $o_{m,1} \oplus \cdots \oplus o_{m,n}$ の計算により，出力 $o_m$ を得ることが出来る．

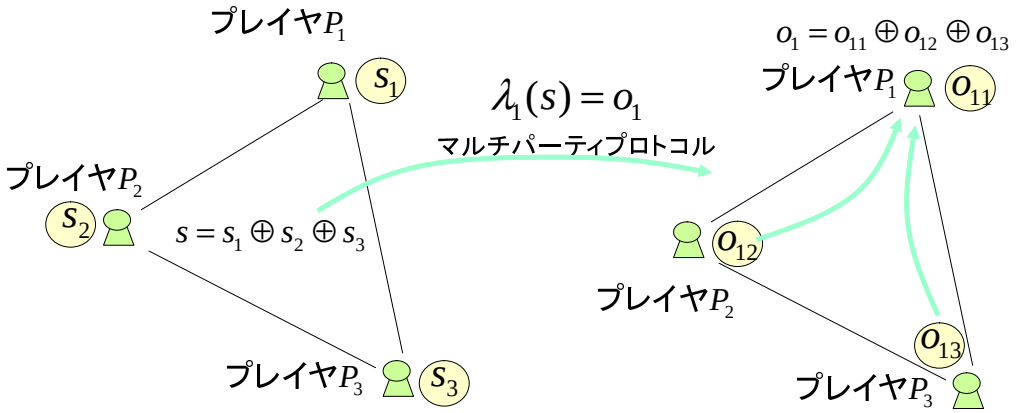


図 7: 出力関数の計算

## 4.3 不正の防御

以下では，不正者が存在する場合，先述のプロトコルによる実装において，起こりうる不正について考察する．

### 4.3.1 不正のレベル

まず，不正の防御方法について述べる．不正の防御方法は以下の３つに分類される．

- **終了後検出可能**
  ゲーム終了時に不正が行われたことと，その不正を行ったプレイヤを突き止められること．
- **即時検出可能**
  不正が行われた時点で不正が行われたことと，その不正を行ったプレイヤを突き止められること．
- **排除可能**
  不正を行うことができないこと．あるいは，不正を行う人がいても，ゲームを続けて進行できること．

このうち，終了後検出の方法はどの不正に対しても同じであるからここで説明する．まず，各プレイヤはゲーム中の送信情報には必ず署名をする．また，プレイヤ間の情報のやりとりを逐一記録しておく．そしてゲームの終了後にそれらを使い，ゲーム開始時からのやりとりの再現をし検証することで，不正を突き止めることが出来る．

このような方法が可能なのはゲームの娯楽性にある．ゲームにおける秘密情報は，専らゲームのためだけの秘密でありゲームが終わればそれをすべて公開しても問題はない．一方で電子投票などでは匿名性の確保のため，終了後も秘密を保持し続けなければならない．これは秘密情報に関するゲームの重要な特徴である．

### 4.3.2 不正の防御可能性

ここでは，3章のゲームモデルのレベルで考えられる不正について例を挙げ，考察する．以下ではゲームのプレイヤ数を3とし，不正者はプレイヤ $P_1$ とする．ここでの議論は，一般の $n$ 人の場合でも成り立つ．

- **不当な入力**

  プレイヤ $P_1$ が，$i_1 \notin I$ を $i_1 = i_{11} \oplus i_{12} \oplus i_{13}$ と分割し，$i_{12}$，$i_{13}$ をそれぞれプレイヤ $P_2, P_3$ に送る．

  この不正は終了後検出可能である．

- **入力の漏洩**

  プレイヤ $P_1$ が自分の入力 $i_1$ を他のプレイヤに送信する．

  この不正は防ぐことはできない．　なぜなら，ネットワーク上では互いに相手が見えないので，ゲームシステムに依存しない別の通信経路を用いて情報を送ることが出来てしまうからである．

- **入力の盗聴**

  プレイヤ $P_1$ がプレイヤ $P_2$ の入力 $i_2$ を直接盗聴する，あるいは，秘密分散された $i_{22}$ と $i_{23}$ を盗聴し，プレイヤ1の持つ $i_{21}$ と合わせることで $i_2$ を得る．

  この不正はネットワークの前提により排除可能である．

- **入力の改ざん**

  プレイヤ $P_1$ がプレイヤ $P_2$ の入力 $i_2$ を $i_2'(\neq i_2)$ に直接すりかえる．あるいは，プレイヤ $P_1$ が受け取った $i_{21}$ を $i_{21}'(\neq i_{21})$ にすりかえ，以降の計算を実行する．

  前者の不正はネットワークの前提により排除可能である．また後者の不正は終了後検出可能である．

- **乱数の盗聴**

  プレイヤ $P_1$ が分散された $r_2, r_3$ を盗聴し，自身のもつ $r_1$ と合わせて $r$ を得る．

この不正はネットワークの前提により排除可能である．

- **乱数の改ざん 1**
  プレイヤ $P_1$ がプレイヤ $P_2, P_3$ の保持している $r_2, r_3$ をそれぞれ $r'_2, r'_3$ $(r'_2 \neq r_2,\ \ r'_3 \neq r_3)$ にすりかえることで，乱数 $r$ を $r' = r'_1 \oplus r'_2 \oplus r'_3$ に改ざんする．
  この不正はネットワークの前提により排除可能である．

- **乱数の改ざん 2**
  プレイヤ $P_1$ が $r_1$ を意図的に選んだとする．このとき，他のプレイヤのうち少なくとも一人 ($P_k$ とする) がランダムに自分の $r_k$ を選べば，$r$ のランダム性は保持される．また，たとえ他のプレイヤが全員意図的に $r_k$ を選んだとしても，お互いの意図がそれぞれ独立であれば，$r$ のランダム性は保持される（お互いの意図が独立でない場合は，その不正は後で述べる結託に含まれるものとする）．
  よって，このような不正は排除可能である．

- **出力の漏洩**
  プレイヤ 1 が自分への出力 $o_1$ を他のプレイヤに送信する．
  この不正は入力の漏洩と同じく防ぐことが出来ない．

- **出力の盗聴**
  プレイヤ 1 がプレイヤ 2 の出力 $o_2$ を直接盗聴する，あるいは，秘密分散された $o_{22}$ と $o_{23}$ を盗聴し，プレイヤ 1 の持つ $o_{21}$ と合わせることで $o_2$ を得る．
  この不正はネットワークの前提により排除可能である．

- **出力の改ざん**
  プレイヤ 1 がプレイヤ 2 の出力 $o_2$ を $o'_2 (\neq o_2)$ に直接すりかえる．あるいは，プレイヤ 1 が受け取った $o_{21}$ を $o'_{21} (\neq o_{21})$ にすりかえ，以降の計算を実行する．
  前者の不正はネットワークの前提により排除可能である．また後者の不正

は終了後検出可能である．

- **ゲーム状態の盗聴**
  プレイヤ 1 が，プレイヤ 2 の持っている $s_2$ と，プレイヤ 3 の持っている $s_3$ を盗聴し，自分の持っている $s_1$ と合わせることで $s$ を得る．この不正はネットワークの前提により排除可能である．

- **ゲーム状態の改ざん**
  プレイヤ 1 がプレイヤ 2 の $s_2$ を $s_2'(\neq s_2)$ に直接すりかえる．あるいは，プレイヤ 1 が自分の $s_1$ を $s_1'(\neq s_1)$ にすりかえ，以降の計算を実行する．
  前者の不正はネットワークの前提により排除可能である．また後者の不正は終了後検出可能である．

- **送信情報の改ざん・否認**
  プレイヤ 1 がプレイヤ 2 に送るべき情報 $a$ を $a'$ にすりかえて送信する．あるいは，プレイヤ 1 がプレイヤ 2 に $a$ を送っておきながら，後でそれを否認する．
  この不正は終了後検出可能である．

- **結託**
  プレイヤ 1 とプレイヤ 2 が結託しているとして，プレイヤ 1 が自分の秘密の情報をプレイヤ 2 に送るとする．
  これは上述の情報の漏洩にあたるので，結託も防ぐことは出来ない．

上述のように，漏洩・結託を除く不正は，送信者の署名と送信情報の逐一記録により少なくとも終了後検出可能であることが言えた．今後は，これらの不正をより高いレベルで防ぐ技術の研究が必要である．

# 第5章　おわりに

本研究では，まず順序機械によりゲームを定式化し，そのモデルにより世にあるゲームを記述し，分類した．また，P2P 型のネットワークにおいてオンラインゲームを実装する際に問題となる点を，このモデルにより考察し，その問題点をマルチパーティプロトコルによる実装によって解決する手法を述べた．そして，この実装において，どのような不正をどのレベルで防ぐことが出来るのかを考察した．結果として，与えられたゲームが我々の定式化においてある種の条件を満たせば，P2P ネットワーク上で安全に実行できるということが分かった．

今後の課題として，起こりうる不正をより高いレベルで防ぐ技術を研究する必要がある．また，個々の技術として安全なものでも，それを組み合わせる時に安全でなくなる場合があり，その考察が必要である．そして，今回考えた一般的な定式化による P2P ネットワーク上でオンラインゲームを実現する方法を，汎用のライブラリとして実装することで，誰でも容易に P2P 型のオンラインゲーム・プログラムが作成できるようにしたい．

# 謝辞

本研究において，多大なるご指導を賜りました岡部寿男教授，宮崎修一助教授，廣瀬勝一講師に感謝致します．また，お世話になっている岡部研究室の皆様に感謝致します．

# 参考文献


[1] J.Smed, T.Kaukoranta and H.Hakonen, “Aspects of Networking in Multiplayer Computer Games”, Proceedings of International Conference on Application and Development of Computer Games in the 21st Century, pp.74-81, 2003.

[2] J.J.Yan and H.J.Choi, “Security Issues in Online Games”, The Electronic Library international journal for the application of technology in information environments, vol.20, no.2, pp.125-133, 2002.

[3] A.Kirmse and C.Kirmse, “Security in online games”, Game Developer, vol.4, no.4, pp.20-28, 1997.

[4] 石川 貴士，石原 進，井手口 哲夫，水野 忠則，“遅延差のあるネットワークにおけるメンバ間公平性保証方式の特性評価”, 情報処理学会論文誌 Vol.42 No.7, 2001.

[5] Nathaniel E. Baughman and Brian Neil Levine, “Cheat-Proof Playout for Centralized and Distributed Online Games”, Proc. IEEE INFOCOM, pp.104-113, 2001.

[6] 爰川 知宏，千綿 伸之，鷲見 卓哉，“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 2002-GN-044, 2002.

[7] Weiliang Zhao, Vijay Varadharajan and Yi Mu, “Fair On-line Gambling”, 16th Annual Computer Security Applications Conference, pp.394-400, 2001.

[8] Weiliang Zhao, Vijay Varadharajan and Yi Mu, “A Secure Mental Poker Protocol Over The Internet”, Proceedings of the Australasian Information Security Workshop, pp.105-109, 2003.

[9] 加藤俊策，宮崎修一，岡部寿男，“不正を検出できるネットワーク軍人将棋”，信学技報，ISEC2003-84, pp.1-6, 2003.

[10] 岡本 龍明，山本 博資，“現代暗号”，産業図書, 1997.

[11] Shimon Even, Oded Goldreich, and Abraham Lempel, “A Randomized Protocol for Signing Contracts”, Communication of the ACM, 28, 6, pp.637-647, 1985.

[12] Oded Goldreich, Silvio Micali, Avi Wigderson, “How to Play Any Mental

Game, or a Completeness Theorem for Protocols with Honest Majority", Proc. of STOC, pp.218-229, 1987.

[13] Bruce Schneier, "Applied Cryptography, Second Edition", John Wiley Sons Inc. , 1995.