

ユーザの結託による不正に強い ハイブリッド P2P 型 MMO ゲームフレームワーク

松本 和馬¹ 岡部 寿男²

概要：

本研究では、P2P 型 MMORPG (Massively Multiplayer Online Role Playing Game) の実現に向けて P2P 型フレームワークを採用することで起こりうる不正を調査し、大きく 3 つに分類した。悪意のある複数のユーザが結託し不正行為を行うことを前提に 3 つの不正の検出可能性やゲームへの影響などを考察した。3 つの不正を防ぐためには、イベントの処理に必要なデータを秘匿したまま計算する必要がある。そこで、スクランブルというデータに何らかの加工をすることでそのデータが本来何を示すデータなのかをわからなくする処理があることを仮定し、それを利用したフレームワークの評価を行った。スクランブルに比較的近い性質を持つ秘密分散を利用した方法を考案し、MMORPG フレームワークでの利用を提案した。提案手法、先行研究、スクランブルを利用したフレームワークについて不正に対する評価を行いそれぞれを比較した。

A Collusion-resilient Hybrid P2P Framework for Massively Multiplayer Online Games

KAZUMA MATSUMOTO¹ YASUO OKABE²

1. はじめに

インターネットを介して大人数が同時に大規模仮想空間内で遊ぶ多人数参加型オンラインゲーム (MMOG: Massively Multiplayer Online Games) は、人気のあるコンテンツで、現在でも様々なゲームが発売されている。MMOG とは「多数のユーザがコンピュータネットワークを介して専用のサーバや他のユーザのクライアントマシン (パソコン、ゲーム機など) と接続し、同じゲーム進行を共有するサービス」と定義され [1]、インターネットを利用したゲームサービスとして一般的なものである。それらの魅力は「数万人規模のユーザが一つの仮想世界に参加し遊ぶ」、「長時間のキャラクター育成やアイテムの収集」、「他のユーザとの協力や協調」とされる。ここから要求要件は「同時に数万人程度のユーザがプレイしリアルタイムに同期」「各

ユーザのデータへの強い改ざん耐性」「公正な競争」と定義できる。

ほとんどの商業 MMOG では「クライアント・サーバ (C/S) 方式」による通信接続方式でシステムを構築する。この方式では、中央管理サーバがゲームの進行処理を行い、クライアントは基本的に、ユーザからの入力受付と、描画や音声出力などユーザへの出力の処理だけを行う。さらに、個々のユーザの個人的なデータ (パーソナルデータ) と大規模仮想空間を構成するための様々なデータ (ワールドデータ) を全て管理者側のストレージで管理する。こうすることで管理者が一元的にセキュリティの確保やデータの管理を行うことが容易であるというメリットが存在する。一方で、C/S 型は負荷がサーバに集中しやすく、最悪の場合過負荷によりサービスが一時停止しかねないスケーラビリティの問題がある。さらにユーザの数が増加するにつれてパーソナルデータ保存のためのストレージの容量が一次関数的に増加する。こういった問題は大規模な設備投資に

¹ 京都大学大学院情報学研究科

² 京都大学学術情報メディアセンター

よって補うことができるが、そのための費用並びに維持管理のためのコストがかさむこととなる。

スケーラビリティの問題を解決するために、C/S 方式に代えて「Peer to Peer(P2P) 方式」を利用した MMOG が研究されている [2][3]。P2P 方式では、各ユーザのゲーム進行処理を各ユーザのクライアントで行い互いにその情報を送信する。P2P 方式により、ゲームの進行処理及びパーソナルデータの保存を、クライアントである各ユーザの端末を利用して行うことができれば、スケーラビリティの問題を解決することができる。一方で、P2P 方式には C/S 方式と違いユーザによる不正が容易であるという致命的な欠点が存在するため、商業的な MMOG で P2P 方式を採用しているものは現状ない。P2P 方式において不正を防ぐ研究も行われているが [4]、C/S 型と同等の堅牢なフレームワークを実現できているものはない。ユーザの不正に対して C/S 型と同じ程度の耐性を持ちつつ、管理者の管理するサーバ及びストレージがスケールする P2P 型のフレームワークが求められている。

先行研究の調査より P2P 型フレームワーク MMOG では「ゲーム進行のイベント処理の改ざん及び妨害」、「本来知りえない情報を知る不正」と「保存されたデータの改ざん」の 3 つの不正が問題であることがわかった。さらに、近年の不正は RMT(Real Money Trade) と呼ばれるゲーム内コンテンツやアカウントの売買を行う規約違反行為の蔓延によって組織的なチートが増大している。したがって悪意あるユーザの結託を前提に、各不正のゲームサービスに対する影響と検出可能性を考察した。本研究ではこの 3 つの不正を防ぐために C/S 型と P2P 型の併用によるハイブリッド型のフレームワークが有効であること、その実現にはユーザに対してデータを秘匿しつつもクライアント上で秘匿したまま計算可能なプロトコル（以下、スクランブル化とよぶ）が必要であることを明らかにした。スクランブルを利用したいくつかのプロトタイプを考案し不正に対する耐性を比較した。さらに、スクランブルの性質に比較的近い秘密分散を利用したフレームワークを提案した。提案手法、先行研究、スクランブルを利用したフレームワークについて不正に対する評価と比較を行い、提案手法が先行研究と比べて一部の重要な不正に対して優れていることを示した。

本論文の構成は以下のようになっている。2 章で関連研究として P2P 型 MMOG についての説明及び不正行為に対する研究について説明する。3 章でスクランブルの概念とスクランブルを利用したフレームワークの不正に対しての耐性を考察する。4 章で秘密分散と秘密分散に基づく秘匿関数計算を利用した提案フレームワークについて説明する。5 章で提案フレームワーク、先行研究、スクランブルを利用したフレームワークの不正に対する耐性を比較し評価する。6 章で結論を述べる。

2. 関連研究

2.1 P2P 型 MMOG の研究について

これまで P2P 型 MMOG の研究において提案されてきたフレームワークは大きく以下の二つに分類される。

- サーバ中心構成法
- ユーザ中心構成法

サーバ中心構成法は、あるユーザに状態データの一部の書き換えを一任して制御する方法である。サーバ役となったユーザは、他のユーザからのアクションメッセージを受け取り、状態データを更新し、それを必要とするユーザに渡す。サーバ役となるユーザはスーパーノードと呼ばれ、ネットワークの帯域が大きいユーザを優先的に選ぶなど特定の条件に従って決定することが多い [5][4]。またサーバ役となったユーザが担当する状態データの割り当ては、MMOG における仮想世界をより小さな細かい範囲（以下：リージョン）に分け、リージョン一つ毎に担当サーバ一台という分け方がされることが多い [5][6]。この方式の欠点として、スーパーノードの負荷及び故障耐性への不安がある点、スーパーノードに情報が集まり不正を行いやすい環境ができ上がる点があげられる。

ユーザ中心構成法は、各ユーザが、アクションを起こした他のユーザからアクションメッセージを直接受け取って、自身が保持する状態データを更新する方法である。この方式は、サーバ中心構成法と違って、すべてのユーザがクライアントとサーバの役割を担う [7]。この方式の欠点として、各ユーザの通信負荷が大きい点、自分自身で保持するデータの改ざんが容易なうえに不正を検知することが難しい点があげられる。

2.2 不正攻撃対策に関しての関連研究

従来の MMOG であっても不正行為（チート）は数多くあり、未だに無くなる気配はない。これらのチートは C/S 型の MMOG であっても防ぐことは出来ない。ゲームの活動ログ及びユーザの通報から管理者が検証し、チートを行ったユーザに対して罰則を与えることでそれを抑制している。数年前までのチート行為は、ゲーム内で有利を得るため、もしくはいたずら目的のものが多かった。しかし、現在では RMT(Real Money Trade) が蔓延し金銭的な利益を得るためのものが多い。中には、グループを組織し効率的にチート行為を行う集団も存在する。こういった増加するチート行為にたいして有効な手段は確立されていない。

P2P 型 MMOG で容易になるであろうチートとして以下のものがある。

- 1 本来知りえない情報を知ることによる不正

- 2 ゲームの進行のイベント処理の改ざん
- 3 ユーザのローカルストレージの直接改ざん

さらに「ゲームの進行のイベント処理の改ざん」のチートは以下のようなものに分類できる。

- 2A 代理計算した結果を、意図して別の値に改竄する攻撃
- 2B 代理計算結果として誤った値を返してゲーム進行を妨害する攻撃
- 2C 代理計算結果を返さないことでゲーム進行を妨害する攻撃

「本来知りえない情報を知ることによる不正」とそのほかの不正では検知可能かどうかという点で異なる。「本来知りえない情報を知ることによる不正」は保存されたデータの内容を見るだけなのでログに記録されることはない。したがって原理的に後で検知することが不可能である。しかしこのチートを行うユーザと行わないユーザが存在すると公平なゲームサービスは提供できない。これは要求要件の「公正な競争」に反する。したがって「本来知りえない情報を知ることによる不正」を防ぐフレームワークを構築し、そのほかのチートは後で検知するという方法が次善的な方法だと考える。本論文では、これら P2P 型 MMOG におけるチートを対象にする。

2.3 多数決を利用した関連研究

遠藤らの研究 [4] では、ユーザが利用するマシンにサーバ機能の一部を委譲してサーバ負荷を分散する構成法において、同じデータを複数のユーザマシンで管理して多数決をとることで「ゲーム進行のイベント処理の改ざん」を防ぎ、同時にユーザマシンの障害に対する冗長性を確保している。サーバ中心構成法に近いが、複数のユーザが同じ状態データを管理しているという点が特徴である。あるリージョンを管理するスーパーノードとなるサーバ役は別のリージョンに存在する PC を操作するユーザとする。こうすることで「本来知りえない情報を知ることによる不正」の影響を低くしている。

しかしながら、チートを行うような悪意あるユーザが SNS などを通じて連絡を取り合うようになりチートが組織的になっている点を考慮すると、不正を防止するための閾値が同じ状態データを管理するスーパーノードの半数という点は不安が残る。また、自 PC が存在するリージョンを管理するスーパーノードにはなれないと言え、特定のユーザは本来知りえない情報を知ることが可能となってしまう。チートを行うような悪意あるユーザがリージョンを跨って連絡を取り合う状況を想定すると、この点は不正の温床となる可能性がある。

2.2 節で挙げたチートに対しての耐性は表 1 のように

なる。

表 1 不正に対する考察：多数決の利用

Cheat	評価	備考
1	×	特定ノードはプレイヤーの情報を知ってしまう
2A	△	過半数が結託すれば可能
2B	△	過半数が結託すれば可能
2C	△	過半数が結託すれば可能
3	△	過半数が結託すれば可能

2.4 ゲーム情報の悪用を防ぐ分散管理手法の提案と評価

山崎らの研究 [8] では、P2P 型 MMORPG でプレイヤーがゲーム情報を管理することにより容易となってしまうチートのうち、ゲーム情報の悪用、すなわち「本来知りえない情報を知ること」による不正を明確化した。さらに、ゲーム内に存在するオブジェクト及びそれに付随する情報を分類した。これを利用して、情報と状態データを分散管理することで「本来知りえない情報を知ること」による不正を防止する手法を提案した。

しかしこの方法では分散管理されたデータの改ざんを許してしまう。また、複数のユーザが結託した場合、情報の悪用が可能になってしまうという欠点が存在する。

前節と同様にチートに対しての耐性は table2 のようになる。

表 2 不正に対する考察：分散管理

Cheat	評価	備考
1	△	一部が結託すれば攻撃可能
2A	×	個人での改ざん可能
2B	×	個人での改ざん可能
2C	△	一部が結託すれば可能
3	△	何のデータ化はわからないが改ざん自体は可能

3. 提案フレームワーク

本章では、2.3 節で述べた多数決を利用する P2P 型 MMOG の構成法を参考に、過半数のユーザが結託したとしても不正を行うことが難しい MMORPG のフレームワークを、C/S 方式を併用したハイブリッド P2P 型で実現する方法を提案する。以下、その基本的な考え方と採用した手法について順を追って詳述する。

3.1 多数決で不正を防ぐ方法

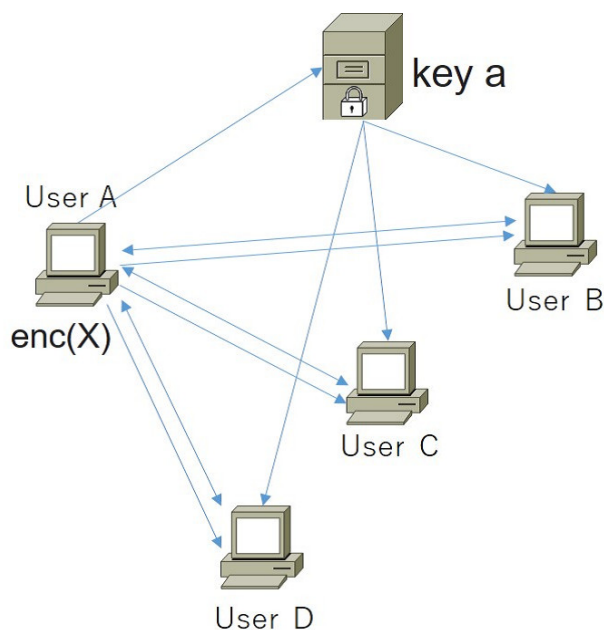


図 1 多数決の導入

図 1 について説明する。 $enc(X)$ はデータ X を暗号化したもの、key a は $enc(X)$ の鍵である。複数台で同じイベント処理を行い、結果を多数決により決定する。こうすることで、個人がイベントの処理またはデータの改ざんを行ったり、結果を返さないことでゲームの進行を停めることを防ぐ。この方式は、2.3 節で述べた多数決を利用する方式を参考に単純化したものである。

- (1) サービスにログインするため管理サーバにアクセス
- (2) $enc(X)$ を管理サーバに送る
- (3) $enc(X)$ を鍵 a で X に復号
- (4) X を B,C,D に送信
- (5) A の入力データを B,C,D に送信
- (6) A の入力データを B,C,D で処理し、結果を B,C,D から A に返す。
- (7) このとき最も多数のものを正しい結果として採用

問題点として、悪人は善人よりも少ないということを前提としていて結託された場合には簡単に不正が行われてしまい信頼性が低い点、依然として本来知りえない情報を知ることによる不正は可能である点が挙げられる。

3.2 スランブル化によるデータの秘匿

本来知りえない情報を知ることによる不正を防ぐために「スランブル化」を考える。ここでいうスランブル化とは、データに何らかの加工をすることでそのデータが本

来何を示すデータなのかをわからなくすることを指す。さらに、スランブル化したままでイベントの処理を行うための計算ができるようなものが望ましい。

図 2 にスランブル化を適用した方式の概念を示す。

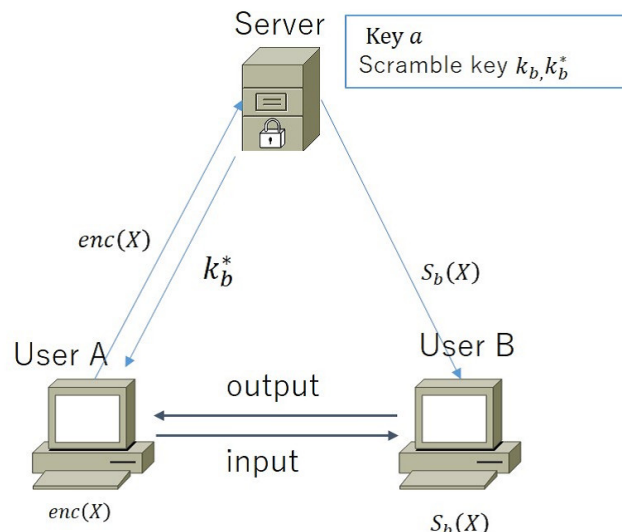


図 2 スランブル化の概念

ユーザが保持している自身のデータ X は、管理サーバにより復号されたのち、動的に生成したスランブル鍵 k_b によりユーザ B に見えないよう変換された上でユーザ B に渡される。ユーザ B は、ユーザ A からスランブル化された入力データを受け取り、ゲームの演算処理を行って、結果をスランブル化されたままユーザ A (プレイヤー) に送る。管理サーバは、入力データのスクランブル化と演算結果のスクランブル解除のための鍵 (部分スクランブル鍵 k_b^* をユーザ A に提示する。

- (1) プレイヤ A はサービスにログインするために管理サーバにアクセス。データ $enc(X)$ を管理サーバに送信
- (2) 管理サーバはスクランブル鍵 k_b 動的に生成、対応する逆スクランブル鍵 k_b^* を A に送信
- (3) 管理サーバは復号した X を k_b でスクランブルした $S_b(X)$ を B に送信
- (4) A の入力データを k_b でスクランブルして B に送信
- (5) A の入力データを B で受け取り、スクランブルされた状態で処理し、結果をスクランブルされたまま B から A に渡す A は逆スクランブル鍵 k_b^* を使って B から送られた結果を復号
- (6) B はデータ $S_b(X)$ 更新結果を定期的に管理サーバに送信 (定期的に改ざんのチェック)
- (7) 管理サーバは更新結果のスクランブルを解除した上で暗号化して A に戻す (データの保存)

こうすることで、ユーザ B による「本来知りえない情報を知ることによる不正」と「データの改ざん」の二つを防

ぎつつ管理サーバの負荷をスケールさせることができる。

3.3 スランブル化と多数決の併用

前節で示した方式では、ユーザ A のデータは代理で計算を行うユーザ B には見えないので、ユーザ B が意図する値にデータや演算結果を改ざんする不正は防げる。しかし、ユーザ B がゲームの進行を妨害する目的で誤った演算結果を返す不正は排除できない。そこで、前節の方式に、2.3 節で述べた多数決を利用する方式を組み合わせることを考える。スランブル化と多数決を併用する方式の基本的な流れを図 3 に示す。

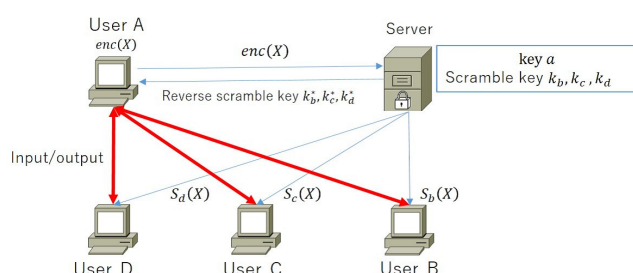


図 3 スランブル化と多数決の併用

サーバは、ユーザ B、ユーザ C、ユーザ D にそれぞれ異なるスランブル鍵でスランブル化されたユーザ A のデータを渡し、代理計算を依頼する。ユーザ A は入力データをスランブル化してユーザ B、ユーザ C、ユーザ D に渡す。代理計算を行うユーザはそれぞれ異なるスランブル鍵でスランブル化されたデータを扱っているので、たとえ過半数が結託したとしても、スランブルを解除したときに同じ値となるような演算結果を返すことができない。それゆえ過半数が結託して悪意を持って誤った値を返す不正を行っても、ユーザ A は多数決を取ることができないことで不正な演算結果が返されたことを検出することができる。

このようなスランブルがあると仮定して 2.2 節で挙げたチートへの耐性を考察した結果が表 3 である。

表 3 不正に対する考察：スランブル+多数決

Cheat	評価	備考
1	○	攻撃不可能
2A	○	攻撃不可能
2B	△	過半数が結託すれば可能
2C	△	過半数が結託すれば可能
3	○	攻撃不可能

3.4 スランブル化に関しての考察

データを加工することでデータの値をわからなくするという点において暗号化はスランブル化の要件を満たす。しかしながら、以上の考察から、スランブル化において

次のような性質が要求される。

- データを変換し、秘匿したまま演算処理が行えること
- ゲームプレイに支障のないレイテンシを維持すること

ここで MMORPG において必要となる演算は以下のものである。

- IF-THEN の条件式
- ゲーム変数の更新
- 加法、乗法

このような秘匿したまま演算を行うことができる暗号化方式として、準同型暗号がある [9][10]。準同型暗号の中でも特に完全準同型暗号では、加法と乗法をどちらも暗号化したまま演算することができる点において、本章で述べたスランブル化において「秘匿したまま演算処理が行える」という要件を満たしている [11]。しかしながら現在のところ、準同型暗号として知られている暗号方式は、いずれもその計算にかかる時間がかかなり大きく、ゲームプレイに支障のないレイテンシを維持するには不適当であると考えられる。

次節では、3.3 節で述べたスランブル化と多数決を併用する方式を修正し、秘密分散と多数決を併用することで、暗号化、復号化ともに準同型暗号に比べ圧倒的に短い実用的な時間でスランブル化と同等のことが実現できることを示す。

4. 秘密分散における秘匿関数計算

4.1 秘密分散法

情報を秘密にするために暗号手法の研究が行われているが、秘密分散法 (Secret Sharing Scheme) とはその手法の一つである。秘密分散法では、秘密を分割し分散することで、秘密の紛失と漏洩への対策を同時に行う [12]。

ある秘密 s が存在し、その秘密保有者 (ディーラ) が避けたいことは秘密 s の紛失と漏洩であるとする。秘密 s の紛失を防ぐために、秘密 s の保有者を複数にすることでその可能性を減らすことが出来る。しかし、それでは漏洩の可能性が上がってしまう。秘密 s の漏洩を防ぐために次のようなことを考える。秘密 s を s_A, s_B, s_C に分割する。分割された値をシェア (share) と呼ぶ。そして、シェア保持者 A, B, C はそれぞれ s_A, s_B, s_C のみを保有する。シェアが全て揃わない限り秘密 s を復元することはできないものとする、秘密 s の漏洩の可能性を減らすことが出来る。しかし、紛失の可能性は上がってしまう。そこで秘密 s を n 個のシェアに分割し、そのうちの k 個が集まれば秘密 s を復元できるが、 $k-1$ 個以下では秘密 s に関して何の情報も得られないような分割方法があるとする。これを (k, n)

閾値秘密分散法と呼ぶ。この時の k を閾値と呼ぶ。

4.2 秘匿関数計算

秘匿関数計算とは、分散して保持されている秘密の値 S_A と S_B を、互いに直接知ることなく、それらの和 $S_A + S_B$ や積 $S_A S_B$ など秘密の値から計算される値を求めることである。

秘密分散における秘匿関数計算は次のように説明される。例として、シェア保持者 A, B, C が分散して秘密の値 s および t をそれぞれ s_A, s_B, s_C および t_A, t_B, t_C として保持しているとする。このとき s と t から計算される値 u の秘密分散 u_A, u_B, u_C について、 u_A を s_A と t_A だけから、 u_B を s_B と t_B だけから、 u_C を s_C と t_C だけから計算するのがこの秘密分散における秘匿計算である。特に $u = s + t$ ならば $u_A = s_A + t_A, u_B = s_B + t_B, u_C = s_C + t_C$ であるとき秘匿関数計算は加法準同型であるという。 $u = s \cdot t$ についての乗法準同型性も同様である。

4.3 中国の剰余定理に基づく秘密分散

4.3.1 中国の剰余定理

中国の剰余定理 (Chinese Remainder Theorem) は以下のように述べられる。

$m_1, m_2, \dots, m_r$ をどの 2 つも互いに素な自然数とすると、任意の整数 $a_1, a_2, \dots, a_r$ に対して、連立合同式

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \vdots \\ x \equiv a_r \pmod{m_r} \end{cases}$$

は整数解 x をもつ。さらに、 $M = m_1 m_2 \dots m_r$ とすると、解は M を法として一意に存在する。

4.3.2 中国の剰余定理を利用した秘密分散の原理

中国の剰余定理における複数の数式を利用すれば解 M が一意に定まる点を秘密分散に利用することができる。

シェアの数を n とする。その互いに素となるような $n+1$ 個の自然数 $m_0 < \dots < m_n$ を考える。 $m_0 \cdot m_{n-k+2} \dots m_n < m_1 \cdot m_2 \dots m_k$ を満たす値を k とする。 $2 < k \leq n (k, n \in \mathbb{N})$ である。 k は閾値である。秘密 S は $\mathbb{Z}/m_0\mathbb{Z}$ を満たす。

$S + \alpha \cdot m_0 < m_1 \dots m_k$ を満たす任意の整数 α を選ぶ。各 m_i を法として $S + \alpha \cdot m_0$ の剰余を取り、それを S_i とする。 $I_i = (S_i, m_i)$ をシェアとする。

ここで $I_{i1}, \dots, I_{ik}$ より以下の連立合同式を作る。

$$\begin{cases} x \equiv S_{i1} \pmod{m_{i1}} \\ x \equiv S_{i2} \pmod{m_{i2}} \\ \vdots \\ x \equiv S_{ik} \pmod{m_{ik}} \end{cases}$$

$m_{i1}, \dots, m_{ik}$ が互いに素なので、中国の剰余定理により $m_{i1} \cdot m_{i2} \dots m_{ik}$ を法として唯一解 S_0 を持つ。秘密 S は m_0 を法とした S_0 の剰余となる。

この手法の有効な点は整数 α の存在によってシェア一つからでは全く秘密 S を推測できない点である [13]。

4.4 秘密分散を利用したスクランブル化

秘密分散とそれにおける秘匿計算を利用することで、3.3 節で述べたスクランブル化と多数決を併用する方式は、以下のように修正される。スクランブル化のためのスクランブル鍵と逆スクランブル鍵は不要となり、管理サーバは秘密分散の方式に基づいてデータを分割し代理計算者に渡せばよい。代理計算された演算結果はユーザ A に返され、ユーザは閾値 k 個以上の結果から演算結果を復元することができる。代理計算を行うユーザは秘密分散されたデータを扱っているの、たとえ閾値以下のユーザが結託したとしても、演算結果を意図するものに改ざんして返すことができない。秘密分散を導入したフレームワークは図 4 のようになる。

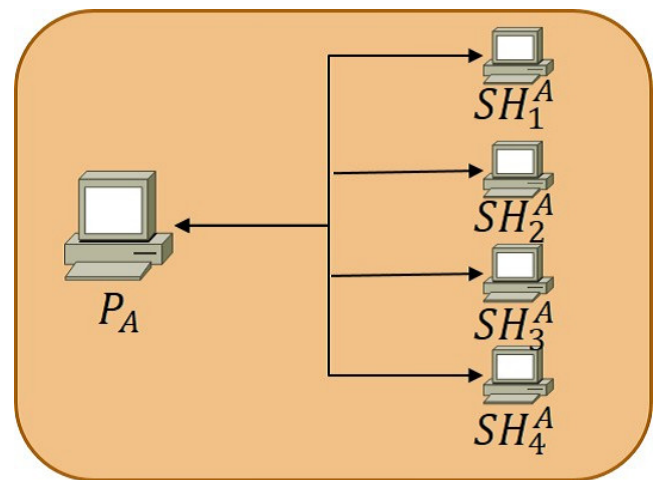


図 4 秘密分散を導入したフレームワーク

中国の剰余定理を利用した秘密分散を導入し、秘匿関数計算によってイベントの処理などを行う提案フレームワークの 2.2 節で挙げられたチート耐性は表 4 のようになる。注 1 について、秘密分散のシェアに誤った値が混じると正しい値を復元することは出来ない。正誤の判定には大きな計算コストがかかってしまう。

表 4 不正に対する考察：提案手法

Cheat	評価	備考
1	$\Delta +$	閾値 K 以上の SH が結託すれば攻撃可能
2A	$\Delta +$	閾値 K 以上の SH が結託すれば攻撃可能
2B	$\times$	注 1
2C	$\Delta -$	N-K 以上のユーザが結託すれば可能
3	$\Delta +$	閾値 K 以上の SH が結託すれば攻撃可能

5. 比較

2.3 節で挙げたチートへの耐性をそれぞれ 2.3, 2.4, 3.3, 4.4 節にて評価した。これらを表 5 にて比較する。

表 5 各手法の比較

Cheat	提案手法	多数決	分散管理	スクランブル
1	$\Delta +$	$\times$	Δ	$\bigcirc$
2A	$\Delta +$	Δ	$\times$	$\bigcirc$
2B	$\times$	Δ	$\times$	Δ
2C	$\Delta -$	Δ	Δ	Δ
3	$\Delta +$	Δ	Δ	$\bigcirc$

1, 2A, 3 について、二つの先行研究よりも閾値の差で優れている。2B, 2C について、耐性はむしろ下がってしまっているがどちらの攻撃も自分が優位になるための攻撃ではない。RMT などの経済目的の不正が流行する現状を考えると使用者の利益にならない攻撃であるこれらの攻撃の耐性は上記の攻撃態勢よりも優先度が下がる。

先行研究の二つの手法より提案手法は重要な 3 つの不正に対して優れていると言える。

6. 結論

本論文では、MMORPG の特徴を精査し、そこから MMORPG の要求要件を「同時に数万人程度のユーザがリアルタイムに同期できる程度のレイテンシ」「パーソナルデータに強い改ざん耐性及び検出可能性」「公正な競争におけるゲーム情報の悪用に対する耐性」とした。C/S 型に比べて P2P 型 MMORPG が普及しない原因は不正が容易である点であることを指摘した。また、P2P で容易になるであろう不正を分類した。さらに、近年の不正行為は経済活動の面が大きく組織的になってきている点を考慮し、先の要求要件の前提として「不正行為者は組織的に結託しうる」ということを指摘し悪意あるユーザに結託されたとしても頑健なフレームワークが必要であると主張した。そして、ハイブリッド P2P を検討することで検出可能性を持たせつつ、要求要件を満たすようなフレームワークを可能にする方法を検討した。関連研究の調査及び先の考察より「データを秘匿したまま演算することが可能」かつ「ゲームプレイに支障のないレイテンシ」という性質が必要であることが分かった。「データを秘匿したまま演算することが

可能」となるようなスクランブル化というものを考案し、それが存在すれば上記の要素を満たす P2P 型 MMORPG が可能であることを示した。そのような性質を満たす可能性がある方法として秘密分散とマルチパーティプロトコルを利用したユーザの結託による不正に強いハイブリッド P2P 型 MMO ゲームフレームワークを提案した。

提案手法、二つの先行研究の方法、スクランブルと多数決を併用した方法それぞれについて分類した不正への耐性を評価した。二つの先行研究よりも 3 つの重要な不正に対してより高い耐性を持っていることを示した。

参考文献

- [1] 中嶋謙互. オンラインゲームを支える技術 一壮大なプレイ空間の舞台裏. 技術評論社, 2011.
- [2] 公原勝彦, 安本慶一, 伊藤実. P2P 環境でのネットワークゲーム向け負荷分散機構の提案. Vol. 2003, No. 19, pp. 79–84, Dec 2003.
- [3] Ignasi Barri, Concepció Roig, and Francesc Giné. Distributing game instances in a hybrid client-server/P2P system to support MMORPG playability. *Multimedia Tools and Applications*, Vol. 75, No. 4, pp. 2005–2029, 2016.
- [4] 遠藤慶一, 川原稔, 高橋豊ほか. 負荷分散型の大規模多人数参加型サービスにおける不正攻撃対策. 情報処理学会論文誌, Vol. 47, No. 4, pp. 1087–1098, 2006.
- [5] Marios Assiotis and Velin Tzanov. A Distributed Architecture for MMORPG. In *Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games*, NetGames '06, New York, NY, USA, 2006. ACM.
- [6] 榎原博之, 吉岡啓, 松崎頼人ほか. 大規模仮想空間における動的領域分割法. 情報処理学会研究報告, Vol. 2012, No. 7, pp. 1–8, 2012.
- [7] 榎本慎太郎, 坂本直志. P2PMMOG のための入退出耐性のある合意プロトコル. 電子情報通信学会技術研究報告. NS, ネットワークシステム, Vol. 111, No. 408, pp. 41–46, 2012.
- [8] 山崎弘太郎, 妙中雄三, 若原恭. P2P Massively Multiplayer Online Role Playing Game におけるゲーム情報の悪用を防ぐ分散管理手法の提案と評価. 電子情報通信学会技術研究報告. IN, 情報ネットワーク, Vol. 111, No. 469, pp. 79–84, Mar 2012.
- [9] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pp. 113–124. ACM, 2011.
- [10] Junfeng Fan and Frederik Vercauteren. Somewhat Practical Fully Homomorphic Encryption. *IACR Cryptology ePrint Archive*, Vol. 2012, p. 144, 2012.
- [11] 神邊 智貴, 佐脇 大貴, 坪井 将洋. 完全準同型暗号を用いたクラウド上の演算機能の実装. Master's thesis, 南山大学 情報理工学部・数理情報学部, 2014.
- [12] 土井洋. 秘密分散法とその応用について. 情報セキュリティ総合科学, Vol. 4, pp. 137–149, 2012.
- [13] Sorin Iftene. General secret sharing based on the chinese remainder theorem. *IACR Cryptology ePrint Archive*, Vol. 2006, p. 166, 2006.