

# メッセージングアプリの普及に基づく 簡単で公平なコイントス

宮原 大輝<sup>1</sup> 水木 敬明<sup>2</sup> 曾根 秀昭<sup>2</sup>

**概要:** 本稿では, LINE に代表されるメッセージングアプリを用いた公平なコイントスの実現手法を提案する. 特に, 送信したメッセージを相手を読んだのかどうかを知らせる既読機能に注目してプロトコルを構成する. 世の中に普及しているメッセージングアプリを対象としているため, そのようなアプリが悪意のある動作を行わないという仮定に基づいて本方式は安全である. 本提案方式は, Blum が提案したコイントス方式等の既存研究に用いられる数学的性質を一切必要としないため, 非専門家でも簡単に理解し実行することができる.

**キーワード:** コイントス, メッセージングアプリ, 既読機能, コミットメント

## 1. はじめに

著者らの研究目的は, 暗号プロトコルを誰でも簡単に実現できる手法を提案することである. 特に, 高校生のような非専門家でも容易に実行可能な方式を示すことで, 暗号を「身近に」感じてもらうことを目指している. 本稿では, 非常に重要な暗号プロトコルであるコイントスに焦点を絞り, その「簡単な」実現方式を提案する. まず始めに, コイントス機能の定義やその既存研究を紹介する. 後で分かるように, コイントスという非常に単純なプリミティブでさえも, 非専門家が気軽に実行するには敷居が高いことに注意されたい.

### 1.1 コイントス

コイントス機能  $\mathcal{F}_{coin}$  は, 1 ビットの乱数  $\sigma \in \{0, 1\}$  を生成し, それを各プレイヤーに与える機能である. あるプロトコルが  $\mathcal{F}_{coin}$  を安全に実現するとは, コイントスの結果である  $\sigma$  をプレイヤーが恣意的に操作できないことを言う. 当然, プレイヤーが同じ場所にいる場合は, 単純に(物理的な)コイントスを行うだけでよいが, プレイヤー同士が物理的に離れている場合, 公平なコイントスの実現方法は自明でない.

1983 年に Blum [1] は, 大きな数の素因数分解が困難であることに安全性の根拠を置いた公平なコイントス方式を提案した. それ以降, コイントスに関する数多くの研究

が成されている. Blum の方式は数学的性質を巧みに用いている反面, その正当性や安全性を理解するには数学的な知識が(多少なりとも)必要である. その実行には計算機(特にプログラム)も必要であるため, 非専門家がすぐに Blum の方式を理解し実行することは難しい.

よりシンプルなコイントス方式として, コミットメントスキーム [2] を用いたコイントスが挙げられる. この方式は 2 節で詳しく説明する. 2 節で分かる通り, この方式は Blum の方式に比べて比較的理解しやすいと考えられるが, やはりその実行には計算機(もしくはコミットメントを提供する特別な web サイト等)が必要不可欠であり, 即座には実行できない.

### 1.2 貢献

1.1 節で説明した通り, 既存方式では, コイントスを気軽には実現できない. そこで本稿では, 近年世の中に普及してきた LINE<sup>i</sup> や Messenger<sup>ii</sup> に代表されるメッセージングアプリを用いることで, 公平なコイントスを簡単に実現できることを示す. 特に, 送信したメッセージを相手を読んだのかどうかを知らせる既読機能に注目してプロトコルを構成する. 後で分かる通り, 提案方式は(スマートフォンと通信環境さえあれば)ほんの数分で実行可能でありその原理もシンプルであるため, 非専門家でも気軽にコイントスを行うことができる. また, 提案方式の変種として,

<sup>1</sup> 東北大学大学院情報科学研究科

<sup>2</sup> 東北大学サイバーサイエンスセンター

<sup>i</sup> <https://line.me/>

<sup>ii</sup> <https://www.messenger.com/>

YouTube<sup>iii</sup> に代表される動画投稿サイトを利用したコイン  
トスも合わせて提案する。この方式では、投稿した動画が  
視聴された回数を表す視聴回数が既読機能の代わりとなっ  
ている。

本方式は世の中に普及しているメッセージングアプリを  
対象としているため、そのようなアプリが悪意のある動作  
を行わないという仮定に基づいて本方式は安全である。す  
なわち、メッセージングアプリがプレイヤーと結託するこ  
とや、プレイヤーがメッセージングアプリに攻撃を加える  
ことはできないとしている。これらの有名アプリは社会イン  
フラになりつつあるほど普及が進んでいるため、アプリ  
に脆弱性が発見された場合にもすぐに対応がなされ、安全  
性が維持されることが期待できる。

本稿の構成を次に示す。2 節では、まずコミットメント  
スキームの定義を紹介し、それを用いたコイントスを示す。  
3 節では、本稿が扱うメッセージングアプリを定義する。4 節  
では、まずメッセージングアプリを用いたコイントスを提  
案し、続いて動画投稿サイトを用いたコイントスを示す。  
5 節では、提案方式を実行する際の注意点を述べる。6 節  
で本稿をまとめる。

### 1.3 関連研究

コイントス機能を簡単に実現する方式として、Salomaa  
による公開鍵暗号の教科書 [3] に、二人が共通して所有し  
ている本（例えば電話帳）と電話機を用いる方式が提案さ  
れている。ここでは詳しく説明しないが、その方式は前述  
のような本が存在すること<sup>iv</sup>を前提としているため、実行  
するにはいささか不便である。

非専門家を対象とした研究はこれまでも多く成されて  
いる。例えば、子どもでもゼロ知識証明を理解できる説明  
を提案した研究 [4] が挙げられる。また、非専門家でも簡  
単に実行できるプロトコルとして、身近な道具であるカー  
ド組のみを用いて秘密計算を行うカードベース暗号 [5–7]  
が近年普及してきている。カードベース暗号は実際に手を  
動かして手軽に秘密計算を行える特徴を持つ。秘密計算の  
コンセプトを教える道具として有用であることを我々は  
オープンキャンパス等で実際に確認している。

カード組以外の身近な道具を用いた研究として、不正開  
封防止シールが貼られた封筒が Moran と Naor によって提  
案されている [8,9]。すなわち、そのような封筒を二者間で  
郵送するプロトコルを行うことで、否認可能なアンケート  
調査 [8] やコミットメント・コイントス [9] を実現でき  
ることが知られている。彼らもその論文で指摘する通り、物  
理的な封筒の性質は非専門家でも容易に理解できるため、

<sup>iii</sup> <https://www.youtube.com/>

<sup>iv</sup> 他にも、その本の内容を誰も暗記していないことや、その本の電  
子版が存在しないといった条件が必要である。詳しくは文献 [3]  
の章 “Cryptographic Protocols Without Computers” を参照  
されたい。

その実装も容易である。しかしながら、“気軽に” 実行でき  
るかどうかは不明である。すなわち、遠く離れた二者間で  
物理的な道具を用いるプロトコルは、郵送による時間的な  
ロスとを当然避けられないため、実際に行うことはいささか  
ためらわれる。

## 2. コミットメントを用いたコイントス

本節では、まずコミットメントの定義を紹介し、続いて  
コミットメントを用いたコイントス方式を紹介する。本稿  
で提案する方式は、2.2 節のコミットメントを用いたコ  
イントス方式とほぼ同じ流れである。本節の内容は文献 [10]  
を参考にしている。

### 2.1 コミットメント

(ビット) コミットメントスキームは、送信者  $S$  と受  
信者  $R$  の二者間で行われるプロトコルであり、コミット  
フェーズとオープンフェーズの二段階で実行される。

**コミットフェーズ** 送信者  $S$  はビット  $b \in \{0,1\}$  を選び、  
乱数  $r$  を生成する。それらを元に、コミットメント  
 $c = \text{com}(b,r)$  を計算して、 $c$  を受信者  $R$  に送信する。  
**オープンフェーズ** 送信者  $S$  は受信者  $R$  に  $b$  と  $r$  を送信  
する。受信者  $R$  は  $\text{com}(b,r)$  を計算し、それが  $c$  と  
等しければ  $b$  を出力する。そうでなければ  $\perp$  を出力  
する。

ここで、 $\perp$  はプロトコルの実行が失敗したことを表す。  
また、 $\text{com}(\cdot, \cdot)$  は送信者  $S$  と受信者  $R$  の間で事前に定め  
られた二入力関数であり、次のような性質を満たす。

**秘匿性** コミットメント  $c$  のみからビット  $b$  を推測するこ  
とは難しい。すなわち、どのような確率的多項式時間  
チューリング機械にとっても、 $\text{com}(0, r_0)$  と  $\text{com}(1, r_1)$   
は確率変数として識別不可能である。

**拘束性**  $c = \text{com}(0, r_0) = \text{com}(1, r_1)$  となる  $(c, r_0, r_1)$  を  
作ることは難しい。

関数  $\text{com}(\cdot, \cdot)$  は、一方向置換を用いて構成できることが  
知られている。

### 2.2 コイントスプロトコル

2.1 節で紹介したコミットメントスキームを用い、1.1 節  
で示したコイントス機能  $\mathcal{F}_{\text{coin}}$  を実現するコイントスプロ  
トコル  $\Pi_{\text{coin}}$  を構成することができる。そのプロトコル  
を次に示す。  $\Pi_{\text{coin}}$  に参加するプレイヤーを  $P_1$  と  $P_2$  と  
する。

- (1)  $P_1$  は 1 ビットの  $b \in \{0,1\}$  と  $n$  ビットの  $r \in \{0,1\}^n$   
を一様ランダムに生成し、 $c = \text{com}(b,r)$  を  $P_2$  に送信  
する。
- (2)  $P_2$  は 1 ビットの  $b' \in \{0,1\}$  を一様ランダムに生成し、  
 $P_1$  に送信する。
- (3)  $P_1$  は  $P_2$  に対し  $c$  をオープンする。すなわち、 $b$  と  $r$

を  $P_2$  に送信する。加えて、 $\sigma = b \oplus b'$  を出力する。

- (4)  $P_2$  は  $c = \text{com}(\sigma, r)$  が成り立つかどうか確認する。成立すれば  $\sigma = b \oplus b'$  を出力し、そうでなければ  $\perp$  を出力する。

正直な  $P_1$  と  $P_2$  に対して、 $\Pi_{\text{coin}}$  が  $\mathcal{F}_{\text{coin}}$  を実現していることは明らかである。 $P_1$  もしくは  $P_2$  が不正を行う場合でも、 $\text{com}(\cdot, \cdot)$  の秘匿性・拘束性から安全であることが証明されている。その厳密な証明は文献 [10] を参照されたい。

今見た通り、コインスプロトコル  $\Pi_{\text{coin}}$  は非常にシンプルであり、 $P_1$  と  $P_2$  間の通信は 3 回である。したがって、このプロトコルの原理を理解し実行することは容易であることが予想される。しかしながら、コミットメントスキームで用いられる関数  $\text{com}(\cdot, \cdot)$  の候補として知られる RSA 関数や有限体上のべき乗関数等を計算機で構成することは決して容易ではない。すなわち、そのような関数を手で実行するための準備が別途必要であり、結局  $\Pi_{\text{coin}}$  を気軽に実行することは難しいことが分かる。

### 3. メッセージングアプリの定義

本節では、提案プロトコルに用いられるメッセージングアプリの機能を定義する。本節における定式化は文献 [9] を参考にしている。厳密な定式化を付録 A.1 に示し、ここでは直感的な説明を行う。

本稿が対象とするメッセージングアプリは、メッセージの送信に加え、既読機能が付いたものである。すなわち、送信されたメッセージの内容を受信者が一たび確認すると、送信者はそのことを（受信者の意思に関係なく）知ることができる。メッセージを受信したこと自体は内容を確認せずに受信者は知ることができることに注意されたい。以上から、本稿が扱うメッセージングアプリには次の四つの機能が存在する。

**送信** この機能を行うのは送信者である。送信者はメッセージを受信者に送信する。この段階では、受信者はメッセージを受信したことだけを知る。後述するように、受信者がそのメッセージの内容を閲覧しない限り、そのメッセージは未読であることを送信者は検証できる。

**閲覧** この機能を行うのは受信者である。受信者は閲覧を行うと、受信したメッセージの内容を知ることができる。この際、メッセージは未読から既読の状態となる。メッセージが削除されていた場合、その内容を知ることができない。

**検証** この機能を行うのは送信者である。送信者は送信したメッセージの内容が未読・既読なのかどうかを知ることができる。

**削除** この機能を行うのは送信者である。送信者は送信したメッセージを削除することで、受信者がその内容を

知ることを防げる。前述したように、メッセージが送信されたと同時に受信者は受信したことを知るため、メッセージを送信したこと自体を送信者は隠すことができない。

メッセージングアプリの機能自体が攻撃されることはないと仮定しているため、上で定義した機能に攻撃者が入り込む余地はない（プロトコルで指定された機能以外の機能を使用する攻撃はあり得る）。しかしながら、送信するメッセージの形態（テキスト・画像・動画）によっては、既読を付けずに内容を確認できることが知られている。このような不正を回避する手法は 5.1 節で詳しく見ていく。送信したメッセージを削除する機能を提案プロトコルで用いることはないが、メッセージングアプリの機能として存在することは妥当である。

## 4. 提案方式

本節では、3 節で定義したメッセージングアプリの機能を使用して、1.1 節で定義した  $\mathcal{F}_{\text{coin}}$  を実現するプロトコルを示す。

### 4.1 提案プロトコル

提案プロトコルは 2.2 節で紹介したコインスプロトコルとほぼ同様の流れを踏む。すなわち、 $P_1$  は 1 ビットの  $b$  をコミットし、 $P_2$  はそれを“予想”する。2.2 節のプロトコルにおいては、コミットメントから元の値を予測できない（秘匿性）ため、 $P_2$  は結局ランダムに  $b$  を予想するしかなかった。提案プロトコルでは、既読機能の付いたメッセージングアプリを介して  $b$  を送信するため、 $P_2$  は  $b$  を確認することができず、結局ランダムに予想することになる。

$\mathcal{F}_{\text{coin}}$  を実現する提案プロトコルを次に示す。

- (1)  $P_1$  は 1 ビットの  $b \in \{0, 1\}$  を一様ランダムに生成し、メッセージングアプリを介して  $P_2$  に  $b$  を送信する。
- (2)  $P_2$  は 1 ビットの  $b' \in \{0, 1\}$  を一様ランダムに生成し、任意の手段（(1) とは別のメッセージングアプリ等）で  $P_2$  に  $b'$  を伝える。
- (3)  $P_1$  はステップ (1) で  $P_2$  に送信した  $b$  が未読であることを検証する。未読であった場合、 $\sigma = b \oplus b'$  を出力し、任意の手段で  $P_2$  に  $\sigma$  を伝える。既読であった場合、 $\perp$  を出力する。
- (4)  $P_2$  は  $b$  を閲覧し、 $\sigma = b \oplus b'$  であることを確かめる。そうである場合、 $\sigma$  を出力する。そうでない場合や、 $b$  が  $P_1$  に削除されていて  $b$  を確認できない場合、 $\perp$  を出力する。

正直な  $P_1$  と  $P_2$  に対し、提案プロトコルが  $\mathcal{F}_{\text{coin}}$  を実現していることは明らかである。次の定理より、 $P_1$  もしくは  $P_2$  が不正を行う場合でも、提案プロトコルは安全であることが分かる。

**定理 1** 提案プロトコルは、1.1 節で定義されたコイン

トス機能  $\mathcal{F}_{coin}$  を安全に実現する.

**証明 1 (sketch)** ここでは証明の概略を示す. 以下では  $P_1$  が不正を行う場合と  $P_2$  が不正を行う場合に分け, 片方に悟られることなくコイントスの出力を恣意的に操作できるのかどうか注目していく.

**$P_1$  が不正を行う場合**  $P_1$  が行動を起こすのはステップ (1) と (3) である. ステップ (1) の時点で  $P_1$  は  $P_2$  が選ぶビット  $b'$  を知り得ないため,  $b$  をランダムに選ぶしかない. ステップ (3) で  $P_1$  は  $\sigma$  を出力するが, ステップ (1) で送信した  $b$  を差し替えることはできない. 結局,  $P_1$  は  $\sigma$  を操作することはできない.

**$P_2$  が不正を行う場合**  $P_2$  が行動を起こすのはステップ (2) と (4) である. ステップ (2) で  $P_2$  は  $P_1$  から送信された  $b$  を閲覧すると, 検証によってそのことが  $P_1$  にばれてしまう. したがって,  $b'$  をランダムに選ばざるを得ない. ステップ (4) で  $P_2$  は  $b$  を閲覧することしかできず, 結局  $P_2$  も  $\sigma$  を操作することはできない.  $\square$

## 4.2 提案プロトコルの変種

本節では, 動画投稿サイトを利用して  $\mathcal{F}_{coin}$  を実現できることを指摘する. プレイヤーはメッセージングアプリを介してビットを送信する代わりに, ビットを動画で表し動画投稿サイトに投稿する. 前節で見たように, 提案方式は既読機能を活用したものであったが, ここでは視聴回数がその役割を担うことになる. すなわち, 投稿した動画の視聴回数が 0 の場合は, その動画の内容は誰にも確認されておらず (未読), 視聴回数が 0 より大きい場合は, 内容が誰かしらに確認された (既読) とみなすことができる. また, 投稿した動画を差し替えることはできないため, コミットしたビットを偽る不正はできない.

付録 A.1 で示したメッセージングアプリ機能  $\mathcal{F}^{(MA)}$  と動画投稿サイト機能は全く同様であるため, その記述は省略する. 加えて, 動画投稿サイトを利用したプロトコルも 4 節で示したプロトコルとほぼ同様であるため, 省略する.

当然, 動画投稿サイトに投稿した動画は, そのままでは誰でも (プロトコルに参加するプレイヤー以外でも) 視聴できるため, 特別な対策<sup>v</sup>が必要である. この対策は 5.2 節で示す.

## 5. 提案方式を実行するにあたって

本節では, 実際に提案方式を実行する際に注意すべき点について述べる. 特に, 既読を付けずに内容を確認する不正を防げるかどうかについて, 実在するメッセージングアプリ同士を比較する.

表 1: メッセージングアプリの比較

メッセージングアプリ	既読機能	安全?
LINE	✓	✓
Messenger	✓	✓
WhatsApp	✓	
Twitter の DM		

### 5.1 メッセージングアプリ

本稿で提案した方式は, 受信者の不正が既読機能によって防がれているものであった. しかしながら, 受信者が既読を付けずに内容を確認できる方法があったとしたら, 提案方式の安全性は崩れる. 実際, メッセージングアプリを介してテキストや画像が送信されると, 受信者は大抵, ポップアップ表示を通して既読を付けることなくその内容を確認できる. すなわち, 送信者が自分のビットをテキストや画像で送信すると, 受信者は送信者に悟られることなくそのビットを知ることができる. ポップアップ表示以外にも, 端末を機内モードにしてメッセージングアプリを起動することで, 送信されたテキストや画像の内容を受信者は確認できる.

上述のような不正は, 送信するメッセージを動画にすることで防ぐことができる. メッセージングアプリを介して送信されたテキストや画像のメッセージはプレイヤーの端末まで送信されるが, 動画はメッセージングアプリを提供するサーバーに留まるため, その動画を再生 (すなわち既読) しない限りその内容を知ることができない.

このような対策手法を用いて実際に提案方式を安全に実現できるメッセージングアプリを表 1 に示す. LINE や Messenger は上述した対策方法が有効であったため, コイントスを安全に実現できる. しかしながら, WhatsApp<sup>vi</sup> に関しては, 端末を機内モードにしても動画を再生することができたため, 提案方式を安全に行うことはできない. 近年, Twitter<sup>vii</sup> のダイレクトメッセージ (DM) 機能にも既読機能が導入されたが, 設定で既読機能の ON/OFF を変更できるため, コイントスを安全に実現できない.

### 5.2 動画投稿サイト

4.2 節で述べた通り, 動画投稿サイトを用いたコイントスは投稿された動画を誰でも視聴可能なため, プロトコルを実行するプレイヤー以外の人動画を視聴してしまう危険性があった. これを防ぐ方法として, 例えば YouTube では投稿した動画の公開範囲を狭めることが挙げられる. YouTube では, 公開範囲を限定公開に設定すると, 動画の URL を知っている者しか視聴できない. したがって, プ

<sup>v</sup> メッセージングアプリでは, 1 対 1 の通信を簡単に確立できる.

<sup>vi</sup> <https://www.whatsapp.com/>

<sup>vii</sup> <https://twitter.com/>

レイヤーは投稿する動画を限定公開とし、その URL をもう一方のプレイヤーだけに教えることで、動画の視聴回数から不正が行われたかどうかを安全に知ることができる。

## 6. おわりに

本稿では、コイントスという極めて単純な暗号プロトコルでさえも非専門家が簡単に実現する手法が無いことをまず指摘した。その上で、近年普及してきたメッセージングアプリを用いることで簡単にコイントスを実現できることを示した。

Future Work として著者らは、秘密計算を“気軽に”実現したいと強く考えている。1.3 節で見た通り、そのような手法であるカードベース暗号が近年注目を浴びているが、これはプレイヤー同士が物理的に同じ空間にいることを前提としている。本稿で提案した方式と同じように、すなわちメッセージングアプリ等の近年普及してきたものを機能として用い、離れたプレイヤー間での気軽な秘密計算（特に論理積）を実現したい。それには、極小モデルの秘密計算 [11] が参考になると考えている。

謝辞 This work was supported by JSPS KAKENHI Grant Number JP17K00001.

## 参考文献

- [1] M. Blum, “Coin flipping by telephone a protocol for solving impossible problems,” SIGACT News, vol.15, no.1, pp.23–27, Jan. 1983.
- [2] G. Brassard, D. Chaum, and C. Crépeau, “Minimum disclosure proofs of knowledge,” Journal of Computer and System Sciences, vol.37, no.2, pp.156–189, 1988.
- [3] A. Salomaa, Public-key Cryptography, Springer-Verlag, Berlin, Heidelberg, 1990.
- [4] J.-J. Quisquater, M. Quisquater, M. Quisquater, M. Quisquater, L. Guillou, M.A. Guillou, G. Guillou, A. Guillou, G. Guillou, and S. Guillou, “How to explain zero-knowledge protocols to your children,” Advances in Cryptology — CRYPTO’ 89 Proceedings, ed. by G. Brassard, vol.10681, pp.628–631, Lecture Notes in Computer Science, Springer New York, New York, NY, 1990.
- [5] B. denBoer, “More efficient match-making and satisfiability the five card trick,” Advances in Cryptology — EUROCRYPT ’89, eds. by J.-J. Quisquater and J. Vandewalle, vol.434, pp.208–217, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 1990.
- [6] C. Crépeau and J. Kilian, “Discreet solitary games,” Advances in Cryptology — CRYPTO’ 93, ed. by D.R. Stinson, vol.773, pp.319–330, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 1994.
- [7] T. Mizuki, M. Kumamoto, and H. Sone, “The five-card trick can be done with four cards,” Advances in Cryptology – ASIACRYPT 2012, eds. by X. Wang and K. Sako, vol.7658, pp.598–606, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2012.
- [8] T. Moran and M. Naor, “Polling with physical envelopes: A rigorous analysis of a human-centric protocol,” Advances in Cryptology - EUROCRYPT 2006, ed. by S. Vaudenay, vol.4004, pp.88–108, Lecture Notes in

- Computer Science, Springer, Berlin, Heidelberg, 2006.
- [9] T. Moran and M. Naor, “Basing cryptographic protocols on tamper-evident seals,” Theoretical Computer Science, vol.411, no.10, pp.1283–1310, 2010.
- [10] O. Goldreich, Foundations of Cryptography: Volume 2, Basic Applications, Cambridge University Press, New York, NY, USA, 2004.
- [11] U. Feige, J. Killian, and M. Naor, “A minimal model for secure computation (extended abstract),” Proceedings of the Twenty-sixth Annual ACM Symposium on Theory of Computing, pp.554–563, STOC ’94, ACM, New York, NY, USA, 1994.

## 付 録

### A.1 メッセージングアプリの定式化

本節では、既読機能を持つメッセージングアプリ機能  $\mathcal{F}^{(MA)}$  を定義する。

機能  $\mathcal{F}^{(MA)}$  は複数の組  $(id, value, creator, holder, state)$  から成る内部テーブルを持つとする。テーブルは状態を表し、 $id$  が一つの組に紐づけられている。以下では、 $value_{id}, creator_{id}, holder_{id}, state_{id}$  はある  $id$  の組に対応する値をそれぞれ表すこととする。テーブルは初め空であり、機能  $\mathcal{F}^{(MA)}$  はプレイヤー  $P_1, \dots, P_n$  の間で次に記述される。

Send  $(value, P_j)$ . プレイヤー  $P_i$  からこのコマンドを受け取ると、機能は  $(id, value, P_i, P_j, \text{unread})$  をテーブルに保存する。加えて、 $P_j$  に  $(\text{Receipt}, id, P_i)$  を出力し、 $P_i$  に  $(\text{Receipt}, id)$  を出力する。

Read  $id$ . プレイヤー  $P_i$  からこのコマンドを受け取ると、機能は  $id$  を持つ組がテーブルに存在し、 $holder_{id} = P_i$  でありかつ  $state_{id} \neq \text{deleted}$  であることを確かめる。もしそうであれば、機能は  $P_i$  に  $(\text{Read}, id, value_{id})$  を出力し、テーブルが保持していた  $id$  の組を  $(id, value_{id}, creator_{id}, holder_{id}, \text{read})$  に置き換える。

Verify  $id$ . プレイヤー  $P_i$  からこのコマンドを受け取ると、機能は  $id$  を持つ組がテーブルに存在し、 $holder_{id} = P_i$  であることを確かめる。もしそうであれば、機能は  $P_i$  に  $(\text{Verify}, state_{id})$  を出力する。

Delete  $id$ . プレイヤー  $P_i$  からこのコマンドを受け取ると、機能は  $id$  を持つ組がテーブルに存在し、 $creator_{id} = P_i$  であることを確かめる。もしそうであれば、機能は  $P_i$  に  $(\text{Delete}, id)$  を出力し、テーブルが保持していた  $id$  の組を  $(id, value_{id}, creator_{id}, holder_{id}, \text{deleted})$  に置き換える。