FairSwapとOptiSwapを一般化したスマートコントラクトに基づく公平交換プロトコル

近藤 貴也[1,a)]　中井 雄士[1]　鈴木 幸太郎[1]

概要：公平交換プロトコルとは，2 者間で電子データを交換する際に，一方の参加者による持ち逃げができないことを保証する暗号プロトコルである．スマートコントラクトに基づく，ファイルとコインの公平交換プロトコルとして FairSwap と OptiSwap が知られている．FairSwap は不正な参加者が存在する場合は OptiSwap よりもラウンド数が小さく効率が良いが，両者が正当な場合は OptiSwap よりも通信量が大きく効率が悪い．OptiSwap は両者が正当な場合は FairSwap よりも通信量が小さく効率が良いが，不正な参加者が存在する場合は FairSwap よりもラウンド数が大きく効率が悪い．つまり，2 つの方式の間には通信量とラウンド数のトレードオフがある．本研究では 2 つの方式を一般化し，通信量とラウンド数をパラメータで調整できる公平交換プロトコルを提案する．提案方式は FairSwap と OptiSwap の両方を含み，さらに 2 つの方式の中間的な効率性を持つ方式も含む．

キーワード：公平交換，暗号通貨，スマートコントラクト，FairSwap, OptiSwap

Fair Exchange based on Smart Contract Generalizing FairSwap and OptiSwap

Takaya Kondo[1,a)]　Takeshi Nakai[1]　Koutarou Suzuki[1]

Abstract: Fair exchange is a cryptographic protocol that enables two parties to exchange their electronic data fairly, i.e., it ensures that no one can steal the other party's item. FairSwap and OptiSwap are fair exchange protocols for files and coins based on smart contracts. While FairSwap is more efficient than OptiSwap regarding the round complexity if a party is corrupted, it has a larger communication overhead if both parties are honest. Also, while OptiSwap is more efficient than FairSwap regarding communication overhead if both parties are honest, it requires larger round number if a party is corrupted. That is, there is a trade-off about round complexity and communication overhead between the two protocols. This work generalizes these two protocols and proposes a fair exchange protocol that allows us to adjust the communication overhead and the number of rounds by a parameter. Our protocol contains FairSwap, OptiSwap, and protocols that have intermediate efficiencies between them.

Keywords: Fair Exchange, Cryptocurrency, Smart Contract, FairSwap, OptiSwap

1. はじめに

1.1 背景

公平交換プロトコルは，2 人の参加者がそれぞれ持つ電子データを公平に交換することを目的とした暗号プロトコルであり，「両方の参加者が相手のデータを得る」または「両方の参加者が相手のデータを得られない」のどちらか一方でプロトコルが終了することを保証する [1], [2]．本稿では特に，ファイル $\boldsymbol{X}$ を保持する Seller と，価格 p のコイン $\mathsf{coins}(p)$ を保持する Buyer との間で行う公平交換を考える．つまり，Buyer が $\mathsf{coins}(p)$ で Seller が持つ $\boldsymbol{X}$ を購

[1] 豊橋技術科学大学，〒 441-8580 愛知県豊橋市天伯町雲雀ヶ丘 1-1
Toyohashi University of Technology, 1-1 Hibarigaoka, Tempakucho, Toyohashi-shi, Aichi, 441-8580, Japan.
[a)] kondo.takaya.ol@tut.jp

入する状況を考える．なお，Buyer は受け取ったファイルが正しいことを検証するための検証式 $\boldsymbol{\phi}$ を保持していることを前提とする．

公平交換プロトコルは，信頼できる第三者などの仮定がなければ実現不可能であることが知られている [3]．この信頼できる第三者として Ethereum などで実装されるスマートコントラクトを用いる研究の流れがある．その代表的な方式の一つが Zero-Knowledge Contingent Payment (ZKCP) [4] [5] である．ZKCP ではオンチェーン上で処理するデータが少ないため，スマートコントラクトの使用に要する手数料について効率が良い．一方で，Buyer による (暗号化された) ファイルの正当性検証にゼロ知識証明方式を用いるため，ファイルサイズが大きい場合は多大な計算コストを要する課題がある．

FairSwap [6] と OptiSwap [7] は，上述の ZKCP の課題を解決するために提案されたスマートコントラクトに基づく公平交換プロトコルである．これらの方式はゼロ知識証明を用いず，マークル木に基づく検証方式 Proof of Misbehaviour (PoM) に基づいて構成される．2 つの方式の構成のアイデアは概ね同様であるが，PoM の実現方法が異なる．それにより 2 つの効率性の間にはギャップがある．FairSwap は不正な動作をする参加者がいる場合（Pessimistic ケース）では OptiSwap よりもラウンド数が小さく効率が良いが，両者が正しく動作した場合（Optimistic ケース）では OptiSwap よりも通信量が大きく効率が悪い．一方で，OptiSwap は，Optimistic ケースでは FairSwap よりも通信量が小さく効率が良いが，Pessimistic ケースでは FairSwap よりもラウンド数が大きく効率が悪い．つまり，2 つの間には通信量とラウンド数のトレードオフがある．

1.2 本研究の貢献

本稿では，FairSwap と OptiSwap を一般化した方式を提案する．提案方式では，公開パラメータによって通信量とラウンド数を調整可能である．提案方式は FairSwap と OptiSwap の両方を含み，さらに 2 つの方式の中間的な効率を持つ方式も含む．そのため，通信量，ラウンド数および手数料について，参加者の要求に応じた公平交換プロトコルを選択して使用できるという利点がある．

2. 準備

正整数 n に対し，$[n] := \{1, \cdots, n\}$ とする．セキュリティパラメータを λ で表し，すべての参加者は λ に関する確率的多項式時間アルゴリズムであるとする．

2.1 回路

述語関数を表す回路 $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_m)$ を有向非巡回グラフとしてモデル化する．$\boldsymbol{\phi}$ の成分 ϕ_i は回路のゲートを表し，$\phi_i = (i, op_i \in \Gamma, I_i \in [i-1]^{\ell_i})$ で定義する．i はそのゲートの識別子を表し，op_i はそのゲートの演算 (の識別子) を表し，I_i はそのゲートの入力ワイヤでつながるゲートの識別子を表す．ここで，Γ は演算の全体集合を表し，ℓ_i は ϕ_i の入力ワイヤの数を表す．$I_i = \emptyset$ を満たす $\boldsymbol{\phi}$ を入力ゲートと呼び，任意の $j \in [m]$ について $m \notin I_j$ を満たすものとし ϕ_m を出力ゲートと呼ぶ．回路 $\boldsymbol{\phi}$ の入力ワイヤの集合を $W_{\mathsf{in}} = \{1, \cdots, n\}$ とし，それ以外のワイヤの集合を $W_{\mathsf{mid}} = \{n+1, \cdots, m\}$ とする．

2.2 マークル木方式

マークル木方式 MT は 3 組の確率的多項式時間アルゴリズム $(\mathsf{MTgen}, \mathsf{MTproof}, \mathsf{MTverify})$ からなる．

$(M_X, r_X) \leftarrow \mathsf{MTgen}(\boldsymbol{X}, Q_X)$： マークル木を生成するアルゴリズム．タプル $\boldsymbol{X} = (x_{i_1}, x_{i_2}, \cdots, x_{i_t})$ と集合 $Q_X = \{i_1, i_2, \cdots, i_t\}$ を入力に取り，タプル $\boldsymbol{X}_{\mathsf{tree}} = (i_1 \parallel x_{i_1}, \cdots, i_t \parallel x_{i_t})$ を作成し，各成分を葉ノードとしたマークル木 M_X とその根ノードの値 r_X を出力する．ここで，任意の $k \in [t-1]$ について，$i_k < i_{k+1}$ を満たすものとする．

$\rho \leftarrow \mathsf{MTproof}(i \parallel x, M)$： $i \parallel x$ がマークル木 M に葉ノードに含まれることの証明を作成するアルゴリズム．要素 $i \parallel x$ とマークル木 M を入力に取り，証明 (Merkle Proof) ρ を出力する．M が n 個の葉ノードを有する場合，ρ は長さ $\lceil \log_2(n) \rceil$ のタプルであり，$i \parallel x$ を値に持つ葉ノードから根ノードまでの経路上に含まれるノードの各兄弟ノードの値とそのパスで構成される．

$1 \text{ or } 0 \leftarrow \mathsf{MTverify}(\rho, i \parallel x, r_X)$： $i \parallel x$ が r_X を根ノードに持つマークル木の葉ノードであるか検証するアルゴリズム．Merkle Proof ρ と識別子 i とマークル木の根 r_X を入力に取り，検証が成功すれば 1 を出力し，検証が失敗すると 0 を出力する．

マークル木方式は，以下の性質を満たすものとする．

- 根ノードの値から葉ノードの値に関する情報を得ることは計算量的に困難である．(hiding 性)
- 葉ノードの集合 X で構成されるマークル木 M に対して，$x' \notin X$ かつ $1 \leftarrow \mathsf{MTverify}(\rho, x', r_X)$ を満たす x を計算することは計算量的に困難である．なお，r_X は M の根ノードを表す．(binding 性)

ハッシュ関数を用いたマークル木の構成において，各ノードの値を決定するハッシュ関数の出力長は衝突困難性を満たすために十分な長さであるとする．なお，ハッシュ関数は Programmability と Observability を持つ Global Random Oracle $\mathcal{H}$ としてモデル化する．(詳細は，[6], [7] 参照)

2.3 公平交換プロトコルの定義

ファイル $\boldsymbol{X}$ を保持する Seller と，検証式 ϕ と価格 p のコイン $\mathsf{coins}(p)$ を保持する Buyer を考える．本稿で扱う公平交換プロトコルでは，Seller は $\mathsf{coins}(p)$ を受け取ること

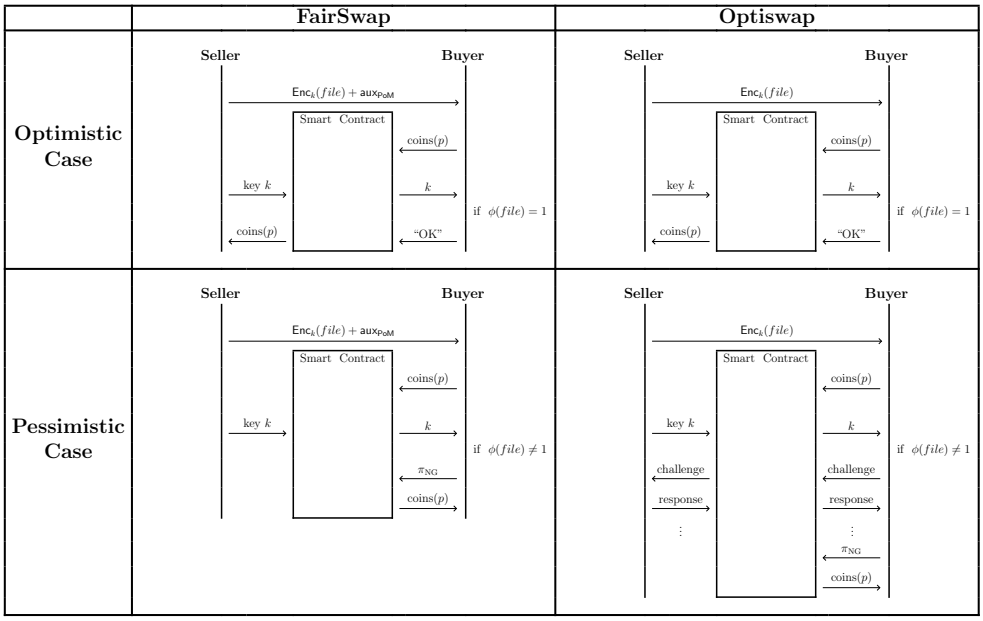

図 1 FairSwap [6] と OptiSwap [7] の概要図

を，Buyer は $\boldsymbol{\phi}(\boldsymbol{X}) = 1$ を満たす $\boldsymbol{X}$ を受け取ることを目的とする．

公平交換プロトコルが安全であるとは，Seller または Buyer を corrupt する任意の確率的多項式時間アルゴリズムの攻撃者に対して，プロトコルが次のどちらか一方を満たした状態でプロトコルが終了することをいう．

- Seller は $\mathsf{coins}(p)$ を得て，かつ，Buyer も $\boldsymbol{\phi}(\boldsymbol{X}) = 1$ を満たす $\boldsymbol{X}$ を得る．
- Seller は $\mathsf{coins}(p)$ を得られず, かつ, Buyer も $\boldsymbol{\phi}(\boldsymbol{X}) = 1$ を満たす $\boldsymbol{X}$ を得られない．

公平交換プロトコルが秘匿性があるとは，参加者以外の第三者に対して $\boldsymbol{X}$ に関する情報が漏洩しないことをいう．

3. 既存方式: FairSwap と OptiSwap の概要

FairSwap と OptiSwap は，Seller と Buyer とは別に Ethereum などで実装されることを想定したスマートコントラクトを信頼できる第三者として用いる．これらの方式は，スマートコントラクトの使用に必要な手数料を抑えるため，ファイル $\boldsymbol{X}$(または，その暗号文) をスマートコントラクトに保持させることをせず，$\boldsymbol{X}$ そのものはローカルのネットワークでやり取りする．さらに，$\boldsymbol{X}$ のサイズが大きい場合でも効率的な処理を達成するため，(ZKCP とは異なり)Buyer による $\boldsymbol{X}$ の正当性検証にゼロ知識証明を用いず，代わりに PoM (Proof of Misbehavior) と呼ばれるマークル木に基づく検証アルゴリズムを採用している．FairSwap と OptiSwap はこの 2 つの設計方針に基づいているという点において共通している．しかし，PoM の実現方法は異なり，それにより 2 つの方式の間には効率性のギャップがある．以下では，そのギャップに焦点を当てて，FairSwap と OptiSwap の概要を示す（図 1 参照）．

Optimistic ケース: まず Optimistic ケースに着目する (図 1 の上段を参照)．FairSwap では，まず Seller からファイルの暗号文に加え，PoM の補助情報として用いるデータ ($\mathsf{aux}_{\mathsf{PoM}}$) を Buyer に送信する (図中では省略しているが，ここで同時に PoM で用いる補助情報をスマートコントラクトに送信する)．なお，具体的には $\mathsf{aux}_{\mathsf{PoM}}$ は検証式 $\boldsymbol{\phi}(\boldsymbol{X})$ を計算する過程の各ワイヤの値を暗号化した値からなる．続いて，Buyer がスマートコントラクトに $\mathsf{coins}(p)$ を入力し，コインの入力を確認した Seller はファイルを復号するための秘密鍵をスマートコントラクトを介して Buyer へ送信する．Buyer は復号結果 $\boldsymbol{X}'$ が $\boldsymbol{\phi}(\boldsymbol{X}') = 1$ を満たすことを確認したら，スマートコントラクトへ "OK" のシグナルを送信し，それをもって Seller は $\mathsf{coins}(p)$ を受け取りプロトコルを終了する．重要な観点として，Optimistic ケースでは PoM は実行されないため，$\mathsf{aux}_{\mathsf{PoM}}$ はこの場合においては一度も使用されない．

OptiSwap も FairSwap とほぼ同様の手順である．唯一の相違点としては OptiSwap では $\mathsf{aux}_{\mathsf{PoM}}$ を生成しない．それにより，Optimistic ケースでは OptiSwap は通信量について FairSwap よりも優れている．

Pessimistic ケース: 次に，Pessimistic ケースに着目する (図 1 の下段を参照)．Seller が不正な参加者であり，Buyer が行うファイルの正当性検証で $\boldsymbol{\phi}(\boldsymbol{X}') = 0$ となったとする．このとき，Buyer は確かに Seller が送信したファイルが不正なものであるということを，スマートコントラクトに証明することで，自身が送信した $\mathsf{coins}(p)$ を取り戻すことができる．(この証明の手続きが PoM に相当する.)

FairSwap では，ファイルが不正であることの証明 π_{NG} を $\mathsf{aux}_{\mathsf{PoM}}$ を用いて生成する．Buyer は π_{NG} をスマートコ

ントラクトに送信することでコインを取り戻すことができる．つまり，FairSwap の PoM は，スマートコントラクトと Buyer の間の 1 ラウンドのみの通信で実現される．

一方で，OptiSwap ではチャレンジレスポンス方式でファイルが不正であることの証明を行う．チャレンジは $\boldsymbol{\phi}$ 中のゲート単位で Buyer が生成し，スマートコントラクトを介して Seller に送信する．また，それに対応する Seller が生成するレスポンスもスマートコントラクトを介して Buyer に送信する．このチャレンジレスポンスは，Seller がレスポンスに失敗するまで繰り返されるため，最悪ケースで $\boldsymbol{\phi}$ のゲート数と同じ数のラウンド数を要する．[*1] したがって，Pessimistic ケースでは，FairSwap はラウンド数 (とそれに伴って生じるスマートコントラクトの手数料) について OptiSwap よりも優れている．

4. 提案方式

Seller を $\mathcal{S}$, Buyer を $\mathcal{B}$ と表記する．$\mathcal{S}$ が保持するファイル $\boldsymbol{X}$ は n 個の同じビット長からなるビット列に分割されているとし，$\boldsymbol{X} = (x_1, \cdots, x_n)$ で表す．$\boldsymbol{\phi} = (\phi_1, \cdots, \phi_m)$ は $\boldsymbol{X}$ の検証アルゴリズムを表す回路とする $(n < m)$．$(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ を IND-CPA 安全な共通鍵暗号方式とする．$(\mathsf{com}, \mathsf{open})$ を hiding 性と binding 性を満たすコミットメント方式とする．[*2] $\mathcal{S}$ と $\mathcal{B}$ の間における直接の通信は，安全な通信路（セキュアチャネル）で行う．

提案方式で用いるスマートコントラクトを $\mathcal{G}_{\mathsf{jc}}$ と表記する．なお，FairSwap と Optiswap では，$\mathcal{G}_{\mathsf{jc}}$ は理想機能として定式化されるが，本稿では簡単のため，$\mathcal{S}$ と $\mathcal{B}$ の間でメッセージを仲介する信頼できる第三者として扱う．$\mathcal{G}_{\mathsf{jc}}$ と参加者との通信はすべて公開され，仲介したメッセージはすべて $\mathcal{G}_{\mathsf{jc}}$ のストレージ領域に保存されるものとする．さらに，各手順の時間制限およびチャレンジレスポンスの回数制限を $\mathcal{G}_{\mathsf{jc}}$ が管理する．

4.1 提案方式の概要

3 節で述べたように，図 1 における $\mathsf{aux}_{\mathsf{PoM}}$ はすべての $\boldsymbol{\phi}(\boldsymbol{X})$ の計算過程における全ワイヤ上の値の暗号文で構成される．提案方式では，PoM の補助情報として送信するこの値の対象ワイヤを，公開パラメータ Q_{pre} として任意に指定可能にする．このとき，$Q_{\mathsf{pre}} = W_{\mathsf{mid}}$ ならば提案方式は FairSwap と一致し，$Q_{\mathsf{pre}} = \emptyset$ ならば OptiSwap と一致する．つまり，提案方式は FairSwap と OptiSwap の一般化である．さらに，Q_{pre} を W_{mid} の真部分集合とすれば 2 つの方式の中間的な効率性を持つ方式を構成できる．

提案方式は，Seller-Init フェーズ，Buyer-Init フェーズ，Key-Reveal フェーズ, File-Extraction フェーズ, Challenge-Response フェーズ, Judge フェーズの 6 つのフェーズからなる．なお，Challenge-Response フェーズおよび Judge フェーズは Pessimistic ケースのみで実行される．

Algorithm 1 $\mathsf{Encode}(\boldsymbol{\phi}, \boldsymbol{X}, \boldsymbol{W}, Q_{\mathsf{pre}}, Q_{\mathsf{post}}, k)$

Input: $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_m)$, $\boldsymbol{X} = (x_1, \cdots, x_n)$, $\boldsymbol{W} = (op_{n+1}(\boldsymbol{X}), \cdots, op_m(\boldsymbol{X}))$,$Q_{\mathsf{pre}} = \{i_1, \cdots, i_t\} \subseteq W_{\mathsf{mid}}$, $Q_{\mathsf{post}} = W_{\mathsf{mid}} \setminus Q_{\mathsf{pre}}$, k

Output: 暗号化されたファイル $\boldsymbol{Z}$, ワイヤの値の暗号文 $\boldsymbol{Z}_{\boldsymbol{Q}_{\mathsf{post}}}$ 鍵のコミットメント (c, d) マークル木の根ノード $r_Z, r_{\mathsf{post}}, r_\phi$

1: $\boldsymbol{Z}_{\boldsymbol{X}} := (z_1 = \mathsf{Enc}_k(x_1), \cdots, z_n = \mathsf{Enc}_k(x_n))$
2: $\boldsymbol{Z}_{\boldsymbol{Q}_{\mathsf{pre}}} := (z_{\min(Q_{\mathsf{pre}})}, \ldots, z_{\max(Q_{\mathsf{pre}})})$
3: $\boldsymbol{Z}_{\boldsymbol{Q}_{\mathsf{post}}} := (z_{\min(Q_{\mathsf{post}})}, \ldots, z_{\max(Q_{\mathsf{post}})})$
4: $\boldsymbol{Z} = \boldsymbol{Z}_{\boldsymbol{X}} \parallel \boldsymbol{Z}_{\boldsymbol{Q}_{\mathsf{pre}}}$
5: $(M_Z, r_Z) \leftarrow \mathsf{MTgen}(\boldsymbol{Z}, [n] \cup Q_{\mathsf{pre}})$
6: $(M_{\mathsf{post}}, r_{\mathsf{post}}) \leftarrow \mathsf{MTgen}(\boldsymbol{Z}_{\boldsymbol{Q}_{\mathsf{post}}}, Q_{\mathsf{post}})$
7: $(M_\phi, r_\phi) \leftarrow \mathsf{MTgen}(\phi, [m])$
8: $(c, d) \leftarrow \mathsf{com}(k)$
9: **return** $(\boldsymbol{Z}, \boldsymbol{Z}_{\boldsymbol{Q}_{\mathsf{post}}}, c, d, r_Z, r_{\mathsf{post}}, r_\phi)$

4.2 提案方式

公開パラメータを $(\lambda, \boldsymbol{\phi}, \mathcal{H}, Q_{\mathsf{pre}}, Q_{\mathsf{post}})$ とする．なお，$Q_{\mathsf{pre}} \subseteq \{n+1, \cdots, m\}$, $Q_{\mathsf{post}} = \{n+1, \cdots, m\} \setminus Q_{\mathsf{pre}}$ とする．$\mathcal{S}$ は $(\boldsymbol{X}, p)$ を初期入力とし，$\mathcal{B}$ は $\mathsf{coins}(p)$ を初期入力としてプロトコルを開始する．

以下に提案方式の手順を示す．なお，手順中の Encode, Extract, ValidateResponse, GenerateProof, Judge は それぞれ Algorithms 1–5 で定義される．

Seller-Init フェーズ

$\mathcal{S}$: 1. $k \leftarrow \mathsf{Gen}(1^\lambda)$
2. $\boldsymbol{\phi}(\boldsymbol{X})$ を計算し，ワイヤ $i \in W_{\mathsf{mid}}$ の値を $\boldsymbol{W} = (op_{n+1}(\boldsymbol{X}), \cdots, op_m(\boldsymbol{X}))$ とする．
3. $(\boldsymbol{Z}, \boldsymbol{Z}_{\boldsymbol{Q}_{\mathsf{post}}}, c, d, r_Z, r_{\mathsf{post}}, r_\phi) \leftarrow \mathsf{Encode}(\boldsymbol{\phi}, \boldsymbol{X}, \boldsymbol{W}, Q_{\mathsf{pre}}, Q_{\mathsf{post}}, k)$
4. $\mathcal{G}_{\mathsf{jc}}$ に $(r_Z, r_{\mathsf{post}}, r_\phi, c, p)$ を送信する．
5. $\mathcal{B}$ に $\boldsymbol{Z}$ を送信する．[*3]

$\mathcal{G}_{\mathsf{jc}}$: 1. $\mathcal{S}$ から $(r_Z, r_{\mathsf{post}}, r_\phi, c, p)$ を受信する．
2. $\mathcal{B}$ に $(r_Z, r_{\mathsf{post}}, r_\phi, p)$ を送信する．

Buyer-Init フェーズ

$\mathcal{B}$: 1. $\mathcal{S}$ から $\boldsymbol{Z}$ を，$\mathcal{G}_{\mathsf{jc}}$ から $(r_Z, r_{\mathsf{post}}, r_\phi, p)$ を受信する．

[*1] 一度のチャレンジで複数のゲートを指定できるため，正確にはより少ないラウンド数での実現も可能である．しかし，その場合スマートコントラクトが処理するデータ量が大きくなるため，手数料がより大きくなるというトレードオフが生じる．

[*2] 厳密には，FairSwap および OptiSwap では安全性証明のため，共通鍵暗号とコミットメント方式は，Global Random Oracle モデル下で具体的な構成が与えられている．本稿では簡単のため，両方式はブラックボックス的に扱うものとする．

[*3] $\boldsymbol{Z}$ に含まれる $\boldsymbol{Z}_{\boldsymbol{Q}_{\mathsf{pre}}}$ が図 1 の $\mathsf{aux}_{\mathsf{PoM}}$ に対応する．

Algorithm 2 Extract$(\boldsymbol{\phi}, \boldsymbol{Z}, Q_{\mathsf{pre}}, k)$

Input: $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_m)$, $\boldsymbol{Z} = (z_1, \cdots, z_n, z_{i_1}, \cdots, z_{i_t})$, $Q_{\mathsf{pre}} = \{i_1, \cdots, i_t\} \subseteq W_{\mathsf{mid}}$, k

Output: 復号したファイル $\boldsymbol{X'}$, $\boldsymbol{\phi}(\boldsymbol{X'})$ を実行した際のワイヤの値 $\boldsymbol{W'}$, 証明 π, ステート情報 state

parse $\boldsymbol{X'} = (x'_1 = \mathsf{Dec}_k(z_1), \cdots, x'_n = \mathsf{Dec}_k(z_n))$
parse $\boldsymbol{W}_{\mathsf{pre}} = (w'_{i_1} = \mathsf{Dec}_k(z_{i_1}), \cdots, w'_{i_t} = \mathsf{Dec}_k(z_{i_t}))$
$\boldsymbol{\phi}(\boldsymbol{X'})$ を実行する．このとき，ワイヤ $i \in W_{\mathsf{mid}}$ の値を $\boldsymbol{W'} = (op_{n+1}(\boldsymbol{X'}), \cdots, op_m(\boldsymbol{X'}))$ とする．
if $\boldsymbol{\phi}(\boldsymbol{X'}) = 1$ **then**
  $\pi = \bot$
  state = Finalize
  return $(\boldsymbol{X'}, \boldsymbol{W'}, \pi, \mathsf{state})$
（$\boldsymbol{\phi}(\boldsymbol{X'}) = 0$ の場合，以下の手順を実行する．）
if $m \in Q_{\mathsf{pre}}$ かつ $w'_m = 0$ **then**
  $\pi_{out} \leftarrow \mathsf{MTproof}(m \parallel z_m, M_Z)$
  $\pi = \pi_{out}$
  state = Judge
  return $(\boldsymbol{X'}, \boldsymbol{W'}, \pi, \mathsf{state})$
else if $w'_i \neq op_i(\boldsymbol{X'})$ かつ $I_i \subseteq [n] \cup Q_{\mathsf{pre}}$ となる $i \in Q_{\mathsf{pre}}$ が存在する **then**
  $\pi_\phi \leftarrow \mathsf{MTproof}(i \parallel \phi_i, M_\phi)$
  $\pi_{out} \leftarrow \mathsf{MTproof}(i \parallel z_i, M_Z)$
  for $j = 1$ **to** ℓ_i **do**
    $\pi^j_{in} \leftarrow \mathsf{MTproof}(I_i[j] \parallel z_{I_i[j]}, M_Z)$
  $\pi = (\pi_\phi, \pi_{out}, \pi^1_{in}, \cdots, \pi^{\ell_i}_{in})$
  state = Judge
  return $(\boldsymbol{X'}, \boldsymbol{W'}, \pi, \mathsf{state})$
else
  $\pi = \bot$
  state = Challenge-Response
  return $(\boldsymbol{X'}, \boldsymbol{W'}, \pi, \mathsf{state})$

Algorithm 3 ValidateResponse$(Q, R, r_{\mathsf{post}})$

Input: Q, R, r_{post}

Output: $y \in \{\mathsf{true}, \mathsf{false}\}$

parse $R = \{(i, z_i, \pi_i)\}_{i \in Q}$
for all $i \in Q$ **do**
  if $(i, \cdot, \cdot) \notin R$ **then**
    return false
  else if $\mathsf{MTverify}(\pi_i, i \parallel z_i, r_{\mathsf{post}}) = 0$ **then**
    return false
return true

2. 以下を計算する．
   - $(M'_Z, r'_Z) \leftarrow \mathsf{MTgen}(\boldsymbol{Z}, [n] \cap Q_{\mathsf{pre}})$
   - $(M'_\phi, r'_\phi) \leftarrow \mathsf{MTgen}(\phi, [n])$
3. $r'_Z \neq r_Z \vee r'_\phi \neq r_\phi$ の場合，アボートしてプロトコルを終了する．
   それ以外の場合，$\mathcal{G}_{\mathsf{jc}}$ に $\mathsf{coins}(p)$ を送信する．

$\mathcal{G}_{\mathsf{jc}}$: 1. $\mathcal{B}$ から $\mathsf{coins}(p')$ を受信する．
2. $p > p'$ の場合，プロトコルを終了する．
   それ以外の場合，$\mathcal{S}$ に $p(= p')$ を送信する．

Key-Reveal フェーズ

$\mathcal{S}$: 1. $\mathcal{G}_{\mathsf{jc}}$ から p を受信する． $\mathcal{G}_{\mathsf{jc}}$ に (k', d) を送信する．

$\mathcal{G}_{\mathsf{jc}}$: 1. $\mathcal{S}$ から (k', d) を受信する．（制限時刻までにメッセージを受信しなかった場合，$\mathsf{coins}(p)$ を Buyer に送信し，プロトコルを終了する．）
2. $\mathsf{Open}(c, k', d) = \mathsf{False}$ の場合，$\mathcal{B}$ に $\mathsf{coins}(p)$ を送信し，プロトコルを終了する．
   $\mathsf{Open}(c, k', d) = \mathsf{True}$ の場合，$\mathcal{B}$ に $k\,(= k')$ を送信する．

File-Extract フェーズ

$\mathcal{B}$: 1. $\mathcal{G}_{\mathsf{jc}}$ から k を受信する．
2. $(\boldsymbol{X'}, \boldsymbol{W'}, \pi, \mathsf{state}) \leftarrow \mathsf{Extract}(\boldsymbol{\phi}, \boldsymbol{Z}, Q_{\mathsf{pre}}, k)$
3. state = Finalize の場合，Key-Reveal フェーズを終了し，$\mathcal{G}_{\mathsf{jc}}$ に “OK” を送信する．その後，$\mathcal{G}_{\mathsf{jc}}$ は $\mathcal{S}$ に $\mathsf{coins}(p)$ を送信し，プロトコルを終了する．
   両参加者が正当な参加者の場合はここまでの手順でプロトコルが終了する．
   state = Judge の場合，$\mathcal{G}_{\mathsf{jc}}$ に π を送信し，Judge フェーズに進む．
   state = Challenge-Response の場合，以下を計算し，Challenge-Response フェーズに進む
   - $Q_{\mathsf{Res}} := Q_{\mathsf{pre}}$
   - $R_{\mathsf{Res}} := \{(i, z_i, \pi_i \leftarrow \mathsf{MTproof}(i \parallel z_i, M_Z)\}_{i \in [n] \cup Q_{\mathsf{Res}}}$
   - $W_{\mathsf{Res}} := \{w'_i = \mathsf{Dec}_k(z_i)\}_{i \in Q_{\mathsf{Res}}}$

   なお，制限時刻までに $\mathcal{G}_{\mathsf{jc}}$ へメッセージを送信しなかった場合，$\mathcal{G}_{\mathsf{jc}}$ は $\mathsf{coins}(p)$ を Seller に送信し，プロトコルを終了する．

Challenge-Response フェーズ (Pessimistic ケース)

以下では，$\mathcal{B}$ と $\mathcal{S}$ の間の通信は $\mathcal{G}_{\mathsf{jc}}$ を仲介するものとして，$\mathcal{G}_{\mathsf{jc}}$ は直近のチャレンジクエリ Q とレスポンスクエリ R を保存する．なお，制限時刻までにチャレンジまたはレスポンスを $\mathcal{G}_{\mathsf{jc}}$ へメッセージを送信しなかった場合，$\mathcal{G}_{\mathsf{jc}}$ は $\mathsf{coins}(p)$ をもう一方の参加者に送信し，プロトコルを終了する．

$\mathcal{B}$: 1. チャレンジクエリ $Q \subseteq Q_{\mathsf{post}}$ を任意に選択する．
2. $\mathcal{G}_{\mathsf{jc}}$ に Q を送信する．

$\mathcal{S}$: 1. $\mathcal{G}_{\mathsf{jc}}$ から Q を受信する．

Algorithm 4 GenerateProof($\boldsymbol{\phi}, \boldsymbol{W'}, W_{\mathsf{Res}}, R_{\mathsf{Res}}, Q_{\mathsf{Res}}$)

Input: $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_m)$, $\boldsymbol{W'}$, W_{Res}, R_{Res}, Q_{Res}

Output: 証明 π

parse $\boldsymbol{W'} = (op_{n+1}(\boldsymbol{X'}), \cdots, op_m(\boldsymbol{X'}))$
parse $W_{\mathsf{Res}} = \{w'_i\}_{i \in Q_{\mathsf{Res}}}$
parse $R_{\mathsf{Res}} = \{(i, z_i, \pi_i)\}_{i \in [n] \cup Q_{\mathsf{Res}}}$
if $w'_m = 0$ **then**
  $\pi_{out} \leftarrow \mathsf{MTproof}(m \parallel z_m, M_Z)$
  $\pi = \pi_{out}$
  return π
else if $w'_i \neq op_i(\boldsymbol{X'})$ かつ $I_i \subseteq [n] \cup Q_{\mathsf{Res}}$ となる $i \in Q_{\mathsf{Res}}$ が存在する **then**
  $\pi_\phi \leftarrow \mathsf{MTproof}(i \parallel \phi_i, M_\phi)$
  $\pi_{out} = \pi_i$
  for $j = 1$ **to** ℓ_i **do**
    $\pi^j_{in} = \pi_{I_i[j]}$
  $\pi = (\pi_\phi, \pi_{out}, \pi^1_{in}, \cdots, \pi^{\ell_i}_{in})$
  return π
else
  $\pi = \perp$
return π

2. レスポンスクエリ
   $R = \{(i, z_i, \pi_i \leftarrow \mathsf{MTproof}(i \parallel z_i, M_{\mathsf{post}}))\}_{i \in Q}$ を計算する．
3. $\mathcal{G}_{\mathsf{jc}}$ に R を送信する．

$\mathcal{B}$: 3. $\mathcal{G}_{\mathsf{jc}}$ から $R = \{(i, z_i, \pi_i)\}_{i \in Q}$ を受信する．
4. $y \leftarrow \mathsf{ValidateResponse}(Q, R, r_{\mathsf{post}})$
5. $y = \mathsf{false}$ の場合，$\mathcal{G}_{\mathsf{jc}}$ に “complain” を送信した後に Judge フェーズに進む．（$y = \mathsf{true}$ の場合，次の手順に進む．）
6. 以下を計算する．
   $W_{\mathsf{Res}} = W_{\mathsf{Res}} \cup \{w'_i = \mathsf{Dec}_k(z_i)\}_{i \in Q}$,
   $R_{\mathsf{Res}} = R_{\mathsf{Res}} \cup \{(i, z_i, \pi_i)\}_{i \in Q}$,
   $Q_{\mathsf{Res}} = Q_{\mathsf{Res}} \cup Q$, $Q_{\mathsf{post}} = Q_{\mathsf{post}} \setminus Q$
7. $\pi \leftarrow \mathsf{GenerateProof}(\boldsymbol{\phi}, \boldsymbol{W'}, W_{\mathsf{Res}}, R_{\mathsf{Res}}, Q_{\mathsf{Res}})$
8. $\pi \neq \emptyset$ の場合，π を $\mathcal{G}_{\mathsf{jc}}$ に送信した後に Judge フェーズに進む．それ以外の場合は，以下を実行する．
   - $Q_{\mathsf{post}} \neq \emptyset$ の場合, Challenge-Response フェーズの先頭の手順に戻る．
   - $Q_{\mathsf{post}} = \emptyset$ またはチャレンジレスポンスの回数制限を超過した場合，$\mathcal{G}_{\mathsf{jc}}$ が $\mathcal{S}$ に $\mathsf{coins}(p)$ を送信し，プロトコルを終了する．

Judge フェーズ（Pessimistic ケース）

$\mathcal{G}_{\mathsf{jc}}$: **$\mathcal{B}$ から π を受信した場合，**

Algorithm 5 Judge($k, \pi, r_Z, r_{\mathsf{post}}, r_\phi$)

Input: k, π, $r_Z, r_{\mathsf{post}}, r_\phi$

Output: $y \in \{\mathsf{true}, \mathsf{false}\}$

parse $\pi = \pi_{out}$ or $\pi = (\pi_\phi, \pi_{out}, \pi^1_{in}, \cdots, \pi^{\ell_i}_{in})$
if $\pi = \pi_{out}$ **then**
  if $\mathsf{MTverify}(\pi_{out}, m \parallel z_m, r_Z) = 1$ かつ $\mathsf{Dec}_k(z_m) \neq 1$ **then**
    return true
  else if $\mathsf{MTverify}(\pi_{out}, m \parallel z_m, r_{\mathsf{post}}) = 1$ かつ $\mathsf{Dec}_k(z_m) \neq 1$ **then**
    return true
  else
    return false
else if $\pi = (\pi_\phi, \pi_{out}, \pi^1_{in}, \cdots, \pi^{\ell_i}_{in})$ **then**
  if $\mathsf{MTverify}(\pi_\phi, i \parallel \phi_i, r_\phi) = 0$ **then**
    return false
  if $\mathsf{MTverify}(\pi_{out}, i \parallel z_i, r_Z) = 0$ かつ $\mathsf{MTverify}(\pi_{out}, i \parallel z_i, r_{\mathsf{post}}) = 0$ **then**
    return false
  for $j = 1$ **to** ℓ_i **do**
    if $\mathsf{MTverify}(\pi^j_{in}, I_i[j] \parallel z_{I_i[j]}, r_Z) = 0$ かつ $\mathsf{MTverify}(\pi^j_{in}, I_i[j] \parallel z_{I_i[j]}, r_{\mathsf{post}}) = 0$ **then**
      return false
  if $\mathsf{Dec}_k(z_i) \neq op_i(\mathsf{Dec}_k(z_{I_i[1]}), \cdots, \mathsf{Dec}_k(z_{I_i[\ell_i]}))$ **then**
    return true
  else
    return false

1. $y \leftarrow \mathsf{Judge}(k, \pi, r_Z, r_{\mathsf{post}}, r_\phi)$
2. $y = \mathsf{true}$ の場合，$\mathcal{B}$ に $\mathsf{coins}(p)$ を送信し，プロトコルを終了する．
   $y = \mathsf{true}$ の場合，$\mathcal{S}$ に $\mathsf{coins}(p)$ を送信し，プロトコルを終了する．

$\mathcal{B}$ から “complain” を受信した場合，

1. $y \leftarrow \mathsf{ValidateResponse}(Q, R, r_{\mathsf{post}})$
2. $y = \mathsf{false}$ の場合，$\mathcal{B}$ に $\mathsf{coins}(p)$ を送信し，プロトコルを終了する．
   $y = \mathsf{true}$ の場合，$\mathcal{S}$ に $\mathsf{coins}(p)$ を送信し，プロトコルを終了する．

4.3 提案方式と FairSwap/OptiSwap との関係

以下では，Seller が不正に動作し $\phi(\boldsymbol{X'}) = 0$ となるような偽造したファイル $\boldsymbol{X'}$ を使用して動作した場合に着目して議論する．このとき，Seller が生成した $\boldsymbol{Z}$ について，以下のどちらかの条件を満たす $i \in W_{\mathsf{mid}}$ が存在することに

注意する．

- $i = m$ であり，$\mathsf{Dec}_k(z_m) = 0$．（つまり，検証式 $\boldsymbol{\phi}$ の演算結果を表す 1-bit が False を示すことを意味する．）
- $\mathsf{Dec}_k(z_i) \neq op_i(\mathsf{Dec}_k(z_{I_i[1]}), \cdots, \mathsf{Dec}_k(z_{I_i[\ell_i]}))$．（つまり，ワイヤ i に対応する演算結果が改ざんされていることを意味する．）

$Q_{\mathsf{pre}} = W_{\mathsf{mid}}$ とした場合の Pessimistic ケースでは，Key-Reveal フェーズで実行する Algorithm 2 で $\mathsf{state} = \mathsf{Judge}$ となる．このとき，File-Extraction フェーズの手順 3 より，必ず Challenge-Response フェーズを介することなく Judge フェーズに進み，Buyer は $\mathsf{coins}(p)$ を取り戻すことができる．したがって，この場合の動作は FairSwap と一致する．

$Q_{\mathsf{pre}} = \emptyset$ とした場合の Pessimistic ケースでは，Key-Reveal フェーズで実行する Algorithm 2 で $\mathsf{state} = \mathsf{Challenge\text{-}Response}$ となる．このとき，File-Extraction フェーズの手順 3 より，必ず Challenge-Response フェーズを仲介して Buyer は $\mathsf{coins}(p)$ を取り戻すこととなる．したがって，この場合の動作は OptiSwap と一致する．

Q_{pre} を W_{mid} の真部分集合のとした場合の Pessimistic ケースでは，Key-Reveal フェーズで実行する Algorithm 2 で $\mathsf{state} = \mathsf{Judge}$ となる場合と，$\mathsf{state} = \mathsf{Challenge\text{-}Response}$ となる場合の 2 通りがある．具体的には，上述の 2 条件どちらかを満たすワイヤ i について，$i \in Q_{\mathsf{pre}}$ ならば $\mathsf{state} = \mathsf{Judge}$，それ以外の場合は $\mathsf{state} = \mathsf{Challenge\text{-}Response}$ となる．Challenge-Response フェーズに進む場合であっても，Q_{pre} に含まれるワイヤについてはチャレンジする必要がないため，Q_{pre} のサイズの分 OptiSwap よりチャレンジレスポンスに要するラウンド数が小さくなる．また，Seller から Buyer に送る PoM の補助情報 $\boldsymbol{Z}$ についても Q_{post} のサイズの分小さくなるため，提案方式は 2 つの方式の中間的な効率を持つといえる．

5. 提案方式の安全性

本節では，両参加者が正当な参加者である場合における安全性および不正な参加者が存在する場合に置いても，安全性を満たすことを (1) $\mathcal{S}$ が不正を行う場合 (2) $\mathcal{B}$ が不正を行う場合 (3) 両参加者が正しく動作する場合 のそれぞれで示す．なお，$Q_{\mathsf{pre}} = W_{\mathsf{mid}}$ および $Q_{\mathsf{pre}} = \emptyset$ の場合はそれぞれ FairSwap と OptiSwap の安全性に帰着できるため，以下では Q_{pre} が W_{mid} の真部分集合であることを前提にする．

5.1 $\mathcal{S}$ が不正を行う場合

不正な $\mathcal{S}$ は $\mathcal{B}$ に正しいファイル $\boldsymbol{X}$ を送信せずに，$\mathsf{coins}(p)$ を得ることを目的とする．この目的を達成するための $\mathcal{S}$ のふるまいとしては，次の 2 つのケースが考えられる．

$\mathcal{S}$-1. 検証式 $\boldsymbol{\phi}'(\neq \boldsymbol{\phi}), \boldsymbol{Z}'(\neq \boldsymbol{Z})$ を偽造する．

$\mathcal{S}$-2. $\boldsymbol{X} \neq \boldsymbol{X}'$ となる $\boldsymbol{X}'$ を偽造する．($\boldsymbol{\phi}(\boldsymbol{X}') = 0$ とする．)

ケース $\mathcal{S}$-1. Seller-Init フェーズで $\mathcal{S}$ が $(M_{\phi'}, r_{\phi'}) \leftarrow \mathsf{MTgen}(\boldsymbol{\phi}', [n])$ を計算し，$\mathcal{G}_{\mathsf{jc}}$ に $r_{\phi'}$ を送信する．Buyer-Init フェーズで $\mathcal{B}$ は $\mathcal{G}_{\mathsf{jc}}$ から $r_{\phi'}$ を受信し，$\boldsymbol{\phi}$ を用いて $(M_{\phi}, r_{\phi}) \leftarrow \mathsf{MTgen}(\boldsymbol{\phi}, [n])$ を計算する．このとき，$\boldsymbol{\phi}$ のマークル木方式の binding 性より，$r_{\phi'} = r_{\phi}$ となる確率は無視可能である．同様に，$\mathcal{S}$ が $\mathcal{B}$ に送信した $\boldsymbol{Z}$ とは異なる $\boldsymbol{Z}'$ でマークル木を計算した場合も，r_Z の binding 性より検知可能である．よって，Buyer-Init の 3 行目の結果より，$\mathcal{B}$ はアボートしてプロトコルを終了する．加えて，$\mathcal{B}$ は $\mathcal{G}_{\mathsf{jc}}$ に $\mathsf{coins}(p)$ を送信しないため，$\mathcal{S}$ は $\mathsf{coins}(p)$ を得られない．したがって，両者は何も得られず，安全性を満たす．

ケース $\mathcal{S}$-2. $\mathcal{B}$ は偽造されたファイル $\boldsymbol{X}'$ の暗号文と Q_{pre} で指定されたワイヤの値の暗号文を含む $\boldsymbol{Z}'$ を Buyer-Init フェーズで受信し，Key-Reveal フェーズで Algorithm 2 を実行する．このとき，$\boldsymbol{\phi}(\boldsymbol{X}') = 0$ であるため，$\boldsymbol{Z}'$ を用いて次に示す $\mathsf{Judge}(k, \pi, r_Z, r_{\mathsf{post}}, r_{\phi}) = \mathsf{true}$ となる条件を満たすこと証明する π を計算する．計算できない場合 Challenge-Response フェーズへ進む．

- $\mathsf{Dec}_k(z'_m) = 0$ かつ，z'_m が r_Z または r_{post} を根ノードに持つマークル木の葉ノードである．
- $\mathsf{Dec}_k(z'_i) \neq op_i(\mathsf{Dec}_k(z'_{I_i[1]}), \cdots, \mathsf{Dec}_k(z'_{I_i[\ell_i]}))$ かつ，ϕ_i が r_{ϕ} を根ノードに持つマークル木の葉ノードであるかつ，$z'_i, z'_{I_i[1]}, \cdots, z'_{I_i[\ell_i]}$ が r_Z または r_{post} を根ノードに持つマークル木の葉ノードである．

Challenge-Response フェーズでは，$\mathcal{S}$ は要求されたワイヤ Q に対して，$\mathsf{ValidateResponse}(Q, R, r_{\mathsf{post}}) = \mathsf{true}$ となる以下の条件を満たしたレスポンスを行う必要がある．

- すべての $i \in Q$ に対して，z'_i が r_{post} の根ノードに持つマークル木の葉ノードである．

Challenge-Response フェーズを繰り返すと，最終的に $\mathcal{B}$ は Algorithm 4 の 8 行目または 13 行目の結果から π を計算できる．このことから，$\mathcal{S}$ が $\mathsf{coins}(p)$ を得るためには，r_{post} または r_Z のマークル木の葉ノードでない z_i が $\mathsf{MTverify}(\rho, i \parallel z_i, r_Z) = 1$ または $\mathsf{MTverify}(\rho, i \parallel z_i, r_{\mathsf{post}}) = 1$ を満たす必要がある．しかし，r_Z, r_{post} のマークル木方式の binding 性より，条件を満たす z_i を計算することは計算量的に困難である．したがって，$\mathcal{G}_{\mathsf{jc}}$ から $\mathcal{B}$ へ $\mathsf{coins}(p)$ が返送されることから，両者は何も得られず，安全性を満たす．

5.2 $\mathcal{B}$ が不正を行う場合

$\mathcal{B}$ は $\mathcal{S}$ に $\mathsf{coins}(p)$ を送信せずに，$\boldsymbol{\phi}(\boldsymbol{X}) = 1$ となる $\boldsymbol{X}$ を得ることを目的とする．この目的を達成するためのふるまいとして次の 3 つのケースが考えられる．

$\mathcal{B}$-1. Buyer-Init フェーズで $\mathcal{G}_{\mathsf{jc}}$ に $\mathsf{coins}(p)$ を送信せずにアボートする．

$\mathcal{B}$-2. File-Extraction フェーズで，$\mathcal{G}_{\mathsf{jc}}$ に何も送信せずにアボートする．

$\mathcal{B}$-3. （$\phi(\boldsymbol{X}') = 1$ を満たすにもかかわらず，）File-Extraction フェーズで $\mathcal{G}_{\mathsf{jc}}$ に Judge または Challenge-Response を送信する．

ケース $\mathcal{B}$-1. $\boldsymbol{Z}$ は共通暗号方式 Enc の暗号文であるため，Enc の IND-CPA 安全性より，暗号文 $\boldsymbol{Z}$ から平文 $\boldsymbol{X}$ に関する情報は漏洩しない．加えて，$\mathcal{B}$ は $\mathcal{G}_{\mathsf{jc}}$ に $\mathsf{coins}(p)$ を送信していないため，$\mathcal{S}$ は $\mathsf{coins}(p)$ を得られない．したがって，両者は何も得られず，安全性を満たす．

ケース $\mathcal{B}$-2. Key-Reveal フェーズ 3 行目より，$\mathcal{B}$ はプロトコルをアボートすると，制限時間を過ぎた後に $\mathcal{G}_{\mathsf{jc}}$ は $\mathcal{S}$ に $\mathsf{coins}(p)$ を送信する．したがって，両者の交換は完了し，安全性を満たす．

ケース $\mathcal{B}$-3. Judge フェーズへ進み，$\mathsf{coins}(p)$ を取り戻すには，$\mathsf{Judge}(k, \pi, r_Z, r_{\mathsf{post}}, r_\phi) = \mathsf{true}$ となる π を $\mathcal{G}_{\mathsf{jc}}$ に送信する必要がある．Algorithm 5 より，true となる条件は次のどちらかを満たすことである．

- $\mathsf{Dec}_k(z_m) = 0$ かつ，z_m が r_Z または r_{post} を根ノードに持つマークル木の葉ノードである．
- $\mathsf{Dec}_k(z_i) \neq op_i(\mathsf{Dec}_k(z_{I_i[1]}), \cdots, \mathsf{Dec}_k(z_{I_i[\ell_i]}))$ かつ，ϕ_i が r_ϕ を根ノードに持つマークル木の葉ノードであるかつ，$z_i, z_{I_i[1]}, \cdots, z_{I_i[\ell_i]}$ が r_Z または r_{post} を根ノードに持つマークル木の葉ノードである．

$\mathcal{S}$ は正当な参加者であることから，$\mathcal{B}$ は Key-Reveal フェーズで上記の条件である，Algorithm 2 の 8 行目または 13 行目を満たす事ができない．同様に Challenge-Response フェーズでチャレンジレスポンスを行った結果，上記の条件である，Algorithm 4 の 4 行目または 8 行目を満たす事ができない．よって，$\mathcal{B}$ は $\mathcal{S}$ から受信した z_i を用いて $\mathsf{Judge}(k, \pi, r_Z, r_{\mathsf{post}}, r_\phi) = \mathsf{true}$ となる π を計算することができない．加えて，r_Z, r_{post} のマークル木方式の binding 性より，z_i を改ざんすることは計算量的に困難である．r_ϕ のマークル木方式の binding 性より，ϕ_i を改ざんすることは計算量的に困難である．このことから，$\mathsf{Judge}(k, \pi, r_Z, r_{\mathsf{post}}, r_\phi) = \mathsf{true}$ となる π を計算することは計算量的に困難である．したがって，両者の交換は完了し，安全性を満たす．

5.3 秘匿性

最後に，プロトコルの通信列から Seller と Buyer 以外の第三者 (仮に Eve とする) に $\boldsymbol{X}$ の情報が漏洩しないことを示す．

Seller と Buyer 間の通信はセキュアチャネルを仮定しているため，Eve が得られる情報は公開パラメータと $\mathcal{G}_{\mathsf{jc}}$ が内部に保持する情報 $(r_Z, r_{\mathsf{post}}, r_\phi, c, k, d, p)$ である．この内，$\boldsymbol{X}$ の情報を含む値は $r_Z, r_{\mathsf{post}}, r_\phi$ であるが，$r_Z, r_{\mathsf{post}}, r_\phi$ のマークル木方式の hinding 性より Eve は $\boldsymbol{X}$ に関する情報を得られない．したがって，Eve は $\boldsymbol{X}$ に関する情報を得られないため，秘匿性を満たす．

以上より，提案方式が安全性を満たすことを確認できた．

注意. 上述の議論には登場しないが，コミットメント方式は UC 安全性の定義の上で安全性証明をする際に寄与する．直観的には，Seller を corrupt した adversary が生成したコミットメント c と暗号文 $\boldsymbol{Z}$ から，simulator がファイル $\boldsymbol{X}$ を抽出できるようにするために用いられる．

6. おわりに

本稿では，スマートコントラクトに基づくファイルとコインの公平交換プロトコル FairSwap と OptiSwap を一般化した方式を提案した．今後の研究課題として，FairSwap と OptiSwap と同様に UC 安全性の定義の下で提案方式に対する形式的な安全性証明を与えることがある．さらに，提案方式の実装評価を行いパラメータによる効率性の変化を解析することも今後の課題である．

謝辞 本研究は JSPS 科研費 JP23K16880 と JST CREST JPMJCR22M1 の助成を受けたものです．

参考文献

[1] Silvio Micali. Simple and fast optimistic protocols for fair electronic exchange. In *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing*, volume 22, pages 12–19, 07 2003.

[2] Alptekin Küpçü and Anna Lysyanskaya. Usable optimistic fair exchange. In *Topics in Cryptology - CT-RSA 2010*, pages 252–267. Springer Berlin Heidelberg, 2010.

[3] Henning Pagnia and Felix C. Gartner Darmstadt. On the impossibility of fair exchange without a trusted third party. In *Technical Report TUD-BS-1999-02*, 1999.

[4] Bitcoin wiki. zero knowledge contingent payment, 2018. `https://en.bitcoin.it/wiki/Zero_Knowledge_Contingent_Payment`.

[5] Matteo Campanelli, Rosario Gennaro, Steven Goldfeder, and Luca Nizzardo. Zero-knowledge contingent payments revisited: Attacks and payments for services. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 229–243, 2017.

[6] Stefan Dziembowski, Lisa Eckey, and Sebastian Faust. Fairswap: How to fairly exchange digital goods. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 967–984, 2018.

[7] Lisa Eckey, Sebastian Faust, and Benjamin Schlosser. Optiswap: Fast optimistic fair exchange. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, pages 543–557, 2020.