Secure Computation with Non-Equivalent Penalties in Constant Rounds

Takeshi Nakai
The University of Electro-Communications, Tokyo, Japan

Kazumasa Shinagawa
Ibaraki University, Ibaraki, Japan
National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

Abstract

It is known that Bitcoin enables to achieve fairness in secure computation by imposing a monetary penalty on adversarial parties. This functionality is called *secure computation with penalties*. Bentov and Kumaresan (Crypto 2014) showed that it could be realized with $O(n)$ rounds and $O(n)$ broadcasts for any function, where n is the number of parties. Kumaresan and Bentov (CCS 2014) posed an open question: "Is it possible to design secure computation with penalties that needs only $O(1)$ rounds and $O(n)$ broadcasts?" In this work, we introduce *secure computation with non-equivalent penalties*, and design a protocol achieving this functionality with $O(1)$ rounds and $O(n)$ broadcasts only. The new functionality is the same as secure computation with penalties except that every honest party receives more than a predetermined amount of compensation while the previous one requires that every honest party receives the same amount of compensation. In particular, both are the same if all parties behave honestly. Thus, our result gives a partial answer to the open problem with a slight and natural modification of functionality.

2012 ACM Subject Classification Security and privacy → Cryptography

Keywords and phrases Secure computation, Fairness, Bitcoin

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2021.5

Funding This work was partially supported by JSPS KAKENHI Grant Numbers JP20J21248 and JP21K17702.

1 Introduction

1.1 Backgrounds

Secure computation enables parties to compute a function whose inputs are their private data [20]. There are several notions of security, such as privacy, correctness, independence of inputs, guaranteed output delivery, and fairness. Fairness requires that at the end of a protocol, either all parties learn the output value or none of them learn it, i.e., no malicious parties receive their output while some honest parties do not receive output. Unfortunately, it is known that secure computation cannot achieve fairness in the standard model if a majority of parties are corrupted [7].

In order to circumvent the impossibility result, there are works to achieve fairness in secure computation by imposing a monetary penalty on aborting parties [17]. We focus on achieving fairness by using decentralized digital currency [3, 2, 1, 4, 13], e.g., Bitcoin [18]. In secure computation on Bitcoin, an aborting party is given a penalty for losing coins, and honest parties are compensated with coins.

Back and Bentov [3] and Andrychowicz, Dziembowski, Malinowski, and Mazurek [2] introduced secure computation on Bitcoin. They studied fair lottery protocols that guarantee any party aborting after learning the result is forced to pay penalties to all other parties. After that, Bentov and Kumaresan [4] formalized such a model of computation as *secure*

3rd International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2021).
Editors: Vincent Gramoli, Hanna Halaburda, and Rafael Pass; Article No. 5; pp. 5:1–5:16
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

computation with penalties. In particular, they defined a *claim-or-refund functionality* $\mathcal{F}^*_{\mathrm{CR}}$ that plays an important role in secure computation with penalties. In $\mathcal{F}^*_{\mathrm{CR}}$, a sender can send coins with a puzzle $\phi_{s,r}(\cdot)$ and a round number τ to a receiver. The receiver gets the coins if he/she reveals a solution w such as $\phi_{s,r}(w) = 1$ in τ, and the sender gets back the coins if the receiver does not publish the solution in τ.

Bentov and Kumaresan [4] showed that secure computation with penalties can be realized for any function in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}^*_{\mathrm{CR}})$-hybrid model, where $\mathcal{F}_{\mathrm{OT}}$ is an ideal functionality of oblivious transfer. Their protocol requires $O(n)$ rounds[1] and $O(n)$ broadcasts, where n is the number of parties and the number of broadcasts is the number of transactions. Kumaresan and Bentov [13] introduced a new functionality $\mathcal{F}^*_{\mathrm{ML}}$ and showed that secure computation with penalties could be realized in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}^*_{\mathrm{ML}})$-hybrid model with only $O(1)$ rounds. However, this protocol requires $O(n^2)$ broadcasts. In the paper, they posed an open problem as follows:

Is it possible to design a fair protocol
that needs only $O(1)$ rounds and $O(n)$ broadcasts?

Related works

Kumaresan, Moran, and Bentov [15] extended secure computation with penalties to the *reactive model* that can also handle multistage functionalities, such as Texas Holdem poker. Also, they defined a new security model for the reactive model and proposed a fair protocol in the reactive model. This paper focuses on the single-stage (i.e., non-reactive) model following the same setting in [4].

Kumaresan and Bentov [14] improved the efficiency of protocols by amortizing the cost over multiple executions. Kumaresan, Vaikuntanathan, and Vasudevan [16] reduced the script complexity of $\phi_{s,r}$. This paper focuses on reducing the number of rounds and the number of broadcasts.

There are several works [11, 6, 8] based on the model with stateful contracts, which is stronger than our model. The model with stateful contracts can be instantiated by an advanced blockchain technique like Ethereum [19], while our model can be instantiated by Bitcoin.

1.2 Our Contributions

We introduce a new functionality, *secure computation with non-equivalent penalties*, which is a slightly relaxed variant of secure computation with penalties. It guarantees that each honest party is compensated with *more than* a predetermined amount of coins, while secure computation with penalties guarantees that every honest party is compensated for the *same* amount of coins. That is, in secure computation with non-equivalent penalties, two honest parties may be compensated with different amounts of coins, although they are at least a predetermined amount.

We show that secure computation with non-equivalent penalties can be realized for arbitrary functions in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}^*_{\mathrm{CR}})$-hybrid model (See Table 1). Our technical contribution is to propose a new *fair reconstruction protocol*, which is a subprotocol of a secure computation

[1] In [5], it is stated that one round should be about an hour on Bitcoin to prevent the double-spending attack. Thus, it implies that a s-round protocol takes about s hours.

Table 1 Comparison of secure computation protocols with (non-equivalent) penalties.

References	# of Rounds	# of Broadcasts	Compensation Amount
Bentov–Kumaresan [4]	$O(n)$	$O(n)$	Equivalent
Kumaresan-Bentov [13]	$O(1)$	$O(n^2)$	Equivalent
This work (Sect. 4)	$O(1)$	$O(n)$	Non-equivalent

protocol with (non-equivalent) penalties, with $O(1)$ rounds and $O(n)$ broadcasts. As a result, we obtain secure computation with non-equivalent penalties with $O(1)$ rounds and $O(n)$ broadcasts by replacing Bentov–Kumaresan's fair reconstruction protocol with ours.

We note that our protocol is equivalent to a protocol achieving secure computation with penalties if all parties behave honestly. Moreover, when we set the least amount of compensation appropriately, malicious behavior is prevented. We believe that our result gives a partial answer to the open problem posed by Kumaresan and Bentov [13].

2 Preliminaries

2.1 Basic Notations

For any positive integer $i \in \mathbb{N}$, $[i]$ denotes the set of integers $\{1, \ldots, i\}$. We denote by n the number of parties in a protocol. We denote by $H \subseteq [n]$ (resp. $C \subseteq [n]$) the set of honest (resp. corrupted) parties. Since each party is either honest or corrupted, it must hold $h + c = n$ for $h := |H|$ and $c := |C|$. We denote by k a security parameter. We assume that all parties are non-uniform probabilistic polynomial-time algorithms in k.

2.2 Secure Computation with Coins

Bentov–Kumaresan [4] introduced a new secure computation model called *secure computation with coins* (SCC) model. It is the same model as the standard model except that entities (i.e., parties, adversaries, ideal functionalities, and an environment) can deal with a non-standard entity called *coins*, which is an atomic entity representing electronic money. Coins are assumed to be having the following properties.

- Coins cannot be duplicated and forged.
- No multiple parties hold the same coin simultaneously.
- Any parties can transfer their coins to other parties freely.
- Each coin is perfectly indistinguishable from one another.

We use the notation $\mathsf{coins}(\cdot)$ to express the amount of coins. If a party owning $\mathsf{coins}(x)$ receives $\mathsf{coins}(y)$ from another party, then the party holds $\mathsf{coins}(x + y)$ as a result.

In the SCC model, some ideal functionalities can deal with coins. We call such a functionality a *special ideal functionality*. These functionalities are described with the superscript $*$, e.g., $\mathcal{F}^*_{\mathrm{xxx}}$. We call an ideal functionality without handling coins a *standard* ideal functionality. Our protocol is realized in the hybrid model where parties have access to a standard functionality $\mathcal{F}_{\mathrm{OT}}$, which is the ideal functionality for oblivious transfer, and a special ideal functionality $\mathcal{F}^*_{\mathrm{CR}}$, described later.

The SCC model follows the real/ideal simulation paradigm as with the standard secure computation model. Let $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k, z)$ denote the output of an environment $\mathcal{Z}$ in the ideal world for realizing an ideal functionality $\mathcal{F}$, where $\mathcal{Z}$ (with an auxiliary input z) is interacting with an ideal adversary $\mathcal{S}$ on security parameter k. Let $\mathrm{HYBRID}^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)$ denote the output of environment $\mathcal{Z}$ in the real (hybrid) world for executing a hybrid protocol π with an ideal functionality $\mathcal{G}$, where $\mathcal{Z}$ is interacting with a real adversary $\mathcal{A}$. The difference with the standard secure computation is that all entities (i.e., parties, adversaries, special ideal functionalities, and an environment) can deal with coins: sending coins, storing coins, and receiving coins.

▶ **Definition 1.** *Let π be a probabilistic polynomial-time n-party protocol and let $\mathcal{F}$ be a probabilistic polynomial-time n-party (standard or special) ideal functionality. We say that π* SCC *realizes $\mathcal{F}$ with abort in the $\mathcal{G}$ hybrid model (where $\mathcal{G}$ is a standard or special ideal functionality) if for every non-uniform probabilistic polynomial-time adversary $\mathcal{A}$, there exists a non-uniform probabilistic polynomial-time adversary $\mathcal{S}$ such that for every non-uniform probabilistic polynomial-time environment $\mathcal{Z}$, two families of random variables $\{\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N},z\in\{0,1\}^*}$ and $\{\mathrm{HYBRID}^{\mathcal{G}}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N},z\in\{0,1\}^*}$ are computationally indistinguishable.*

2.3 Special Ideal Functionalities

2.3.1 Claim-or-refund functionality $\mathcal{F}^*_{\mathrm{CR}}$

This functionality $\mathcal{F}^*_{\mathrm{CR}}$ [4] can be seen as an analogue of puzzles with bounty. Roughly speaking, for a puzzle $\phi_{s,r}$ with coins submitted by a sender, a receiver gets the coins if and only if he/she submits a solution w of the puzzle (i.e., $\phi_{s,r}(w)=1$). $\mathcal{F}^*_{\mathrm{CR}}$ consists of three phases: deposit, claim, and refund. In the deposit phase, a sender P_s sends to a receiver P_r "conditional" coins together with a circuit $\phi_{s,r}$. The coins also have a round number τ specified by the sender. In the claim phase, the receiver P_r claims to receive the coins. P_r can receive the coins only if he/she broadcasts the witness w of $\phi_{s,r}$ (i.e., $\phi_{s,r}(w)=1$) in τ. Note that the witness w published in the claim phase is made public to all parties. In the refund phase, if P_r does not claim in τ, then the coins are refunded to the sender P_s. See Algorithm 1 for a formal description of $\mathcal{F}^*_{\mathrm{CR}}$.[2] At least one broadcast is necessary to realize $\mathcal{F}^*_{\mathrm{CR}}$ on Bitcoin. Thus, the number of calling $\mathcal{F}^*_{\mathrm{CR}}$ corresponds to the number of broadcasts.

We call the message in the deposit phase a deposit transaction. We use the following "arrow" notation to denote the deposit transaction for the sender P_s and the receiver P_r.

$$P_s \xrightarrow[c,\tau]{w} P_r$$

After making an arrow from P_s to P_r as above (i.e., after the deposit phase), P_r can claim to receive $\mathsf{coins}(c)$ only if he/she publishes the witness w in round τ. $\mathsf{coins}(c)$ is refunded back to the original holder P_s if P_r does not publish w in τ.

2.3.2 Secure computation with penalties $\mathcal{F}^*_f$

This functionality $\mathcal{F}^*_f$ is the same as the standard secure function evaluation except that aborting parties are forced to pay penalties [4]. In principle, it guarantees the following properties:

- no honest party pays any penalty, and
- if a party aborts after learning the output value and does not tell the value to the other parties, then every party who does not learn the value is compensated with coins.

[2] [5, 14] show how to realize $\mathcal{F}^*_{\mathrm{CR}}$ using Bitcoin.

Algorithm 1 Claim-or-refund functionality $\mathcal{F}^*_{\mathrm{CR}}$ [4].

```
Setup The session identifier is $sid$. Running with parties $P_1, \dots, P_n$ and an ideal adversary $\mathcal{S}$.
Deposit phase Receiving $(\mathrm{deposit}, sid, ssid, s, r, \phi_{s,r}, \tau, \mathsf{coins}(c))$ from $P_s$, perform the following process.
    1) Record the message $(\mathrm{deposit}, sid, ssid, s, r, \phi_{s,r}, \tau, c)$
    2) Send all parties $(\mathrm{deposit}, sid, ssid, s, r, \phi_{s,r}, \tau, c)$
        - Ignore any future messages with the same $ssid$ from $P_s$ to $P_r$.
Claim phase Receiving $(\mathrm{claim}, sid, ssid, s, r, \phi_{s,r}, \tau, c, w)$ from $P_r$ in round $\tau$, perform the following process.
    1) Check the two conditions:
        - $(\mathrm{deposit}, sid, ssid, s, r, \phi_{s,r}, \tau, c)$ was recorded,
        - $\phi_{s,r}(w) = 1$.
    2) If both checks are passed, perform the following process:
        2-i) send $(\mathrm{claim}, sid, ssid, s, r, \phi_{s,r}, \tau, c, w)$ to all parties,
        2-ii) send $(\mathrm{claim}, sid, ssid, s, r, \phi_{s,r}, \tau, \mathsf{coins}(c))$ to $P_r$,
        2-iii) delete the record $(\mathrm{deposit}, sid, ssid, s, r, \phi_{s,r}, \tau, c)$.
Refund phase In $\tau + 1$, if the record $(\mathrm{deposit}, sid, ssid, s, r, \phi_{s,r}, \tau, c)$ was not deleted, then perform the following process:
    1) send $(\mathrm{refund}, sid, ssid, s, r, \phi_{s,r}, \tau, \mathsf{coins}(c))$ to $P_s$,
    2) delete the record $(\mathrm{deposit}, sid, ssid, s, r, \phi_{s,r}, \tau, c)$.
```

See Algorithm 2 for a formal description of $\mathcal{F}^*_f$. The parameters q and d specify the amounts of coins. At the beginning of the protocol, each party submits $\mathsf{coins}(d)$ together with input x_i. If a party aborts after learning the output and does not tell the value to the other parties, then $\mathcal{F}^*_f$ gives $\mathsf{coins}(q)$ to every party who does not learn the output as compensation. Then, it is important note that the compensation amount is always q for any parties.

H is a set of honest parties and $H' \subseteq H$ is a subset chosen by $\mathcal{S}$, which represents parties who are compensated. At first glance, it is somewhat strange that $\mathcal{S}$ chooses a subset of honest parties. The reason why H' is needed is that there are two types of aborting in secure computation with abort. The first one is that an adversary aborts after obtaining the output and thus honest parties cannot obtain the outputs. In this case, $\mathcal{S}$ chooses $H' = H$ and all honest parties are compensated with $\mathsf{coins}(q)$ although the output is stolen by the adversary. The second one is that an adversary aborts before obtaining the output so the protocol just terminates. In this case, $\mathcal{S}$ chooses $H' \subsetneq H$ (possibly empty) and the parties in H' are compensated with $\mathsf{coins}(q)$.[3]

2.4 Non-malleable secret sharing with public verifiability and public reconstructibility

A *non-malleable secret sharing scheme with public verifiability and public reconstructibility* (in short, $\mathsf{pubNMSS}$) [4] is a variant of non-malleable secret sharing scheme. The share algorithm of $\mathsf{pubNMSS}$ takes a secret s as input, generates "tag-token" pairs $(\mathsf{Tag}_i, \mathsf{Token}_i)_{i \in [n]}$, and

[3] H'' is required for a technical reason in order to prove the security. See [4] for a detail. In order to prove the security of our protocol, our new functionality follows the same strategy.

Algorithm 2 Secure computation with penalties $\mathcal{F}^*_f$ [4].

```
Setup The session identifier is $sid$. Running with parties $P_1, \ldots, P_n$, and an ideal adversary
  $\mathcal{S}$ that corrupts parties $\{P_i\}_{i \in C}$. Let $d$ be a parameter representing the safety deposit,
  and let $q$ denote the penalty amount.
Input phase Wait to receive the following messages.
  - $(\text{input}, sid, ssid, i, x_i, \mathsf{coins}(d))$ from $P_i$ for all $i \in H$
  - $(\text{input}, sid, ssid, \{y_i\}_{i \in C}, H', \mathsf{coins}(h'q))$ from $\mathcal{S}$, where $H' \subseteq H$ and $h' = |H'|$
Output phase Perform the following process.
  1) Send $(\text{return}, sid, ssid, \mathsf{coins}(d))$ to each $P_i$ for $i \in H$.
  2) Compute $(y_1, \ldots, y_n) \leftarrow f(x_1, \ldots, x_n)$.
     - if $h' = 0$, then send message $(\text{output}, sid, ssid, y_i)$ to $P_i$ for $i \in H$, and terminate.
     - If $0 < h' < h$, then send $(\text{extra}, sid, ssid, \mathsf{coins}(q))$ to $P_i$ for each $i \in H'$, and
       terminate, where $h := |H|$.
     - If $h' = h$, then send message $(\text{output}, sid, ssid, \{y_i\}_{i \in C})$ to $\mathcal{S}$.
  3) If $\mathcal{S}$ returns $(\text{continue}, sid, ssid, H'')$, where $H'' \subseteq H$, then perform the following
     process:
       3-i) send $(\text{output}, sid, ssid, y_i)$ to $P_i$ for all $i \in H$,
      3-ii) send $(\text{payback}, sid, ssid, \mathsf{coins}((h - h'')q))$ to $\mathcal{S}$ where $h'' = |H''|$,
     3-iii) send $(\text{extrapay}, sid, ssid, \mathsf{coins}(q))$ to $P_i$ for each $i \in H''$.
  4) Else if $\mathcal{S}$ returns $(\text{abort}, sid, ssid)$, send $(\text{penalty}, sid, ssid, \mathsf{coins}(q))$ to $P_i$ for all $i \in H$.
```

outputs Token_i and $(\mathsf{Tag}_1, \ldots, \mathsf{Tag}_n)$ to each party P_i. The parties can reconstruct s by collecting all n tokens. For all $i \in [n]$, the parties can verify if the published Token_i is valid with Tag_i. The tag-token pairs have the following properties:

- all tags $(\mathsf{Tag}_1, \ldots, \mathsf{Tag}_n)$ leak no information about s,
- any sets of $t(< n)$ tokens leak no information about s,
- for any $i \in [n]$, the adversary cannot generate $\mathsf{Token}'_i (\neq \mathsf{Token}_i)$ such that $(\mathsf{Tag}_i, \mathsf{Token}'_i)$ is a valid tag-token pair.

A $\mathsf{pubNMSS}$ scheme can be obtained from the *honest-binding commitment*, which can be constructed from one-way functions [9]. Tag_i is an (honest-binding) commitment that is computed by a secret share sh_i and a randomness r_i as input, and $\mathsf{Token}_i := (sh_i, r_i)$. Namely, the parties can verify if the published $\mathsf{Token}'_i = (sh'_i, r'_i)$ is valid by comparing Tag_i and the commitment whose input is sh'_i and r'_i. In the following discussions, this verification corresponds to $\phi_{s,r}$ in $\mathcal{F}^*_{\text{CR}}$ executions.

3 Existing Protocol for secure computation with penalties

In this section, we introduce Bentov–Kumaresan's protocol [4] for secure computation with penalties in the $(\mathcal{F}_{\text{OT}}, \mathcal{F}^*_{\text{CR}})$-hybrid model.

3.1 Bentov–Kumaresan's Protocol

For a function f, an *augmented function* denoted by $\hat{f}$ is defined by a function that takes an input x and distributes secret shares of the output value $f(x)$. The underlying secret sharing scheme is non-malleable secret sharing with publicly verifiability and publicly reconstructibility (Section 2.4). Thus the augmented function $\hat{f}$ outputs a token Token_i (i.e., a share of $f(x)$) and a set of tags $(\mathsf{Tag}_1, \ldots, \mathsf{Tag}_n)$ to party P_i.

Bentov–Kumaresan's protocol proceeds as follows:

(i) The parties execute a secure computation protocol for $\hat{f}$, and then each party P_i obtains a token Token_i of $f(x)$ and a set of tags $(\mathsf{Tag}_1, \dots, \mathsf{Tag}_n)$. (Note that this is the standard computation without Bitcoin).

(ii) For the reconstruction of tokens, the parties execute the *fair reconstruction protocol*, where each party P_i is forced to broadcast a token Token_i. The validity of the submitted token Token_i is verified with the tag Tag_i. (Note that this computation is based on Bitcoin).

It is well known that the OT functionality $\mathcal{F}_{\mathrm{OT}}$ is sufficient to achieve secure computation for any standard functionality [12, 10]. Moreover, this can be performed in constant rounds [10]. Therefore, the secure computation stage (i) is performed in constant rounds in the $\mathcal{F}_{\mathrm{OT}}$-hybrid model.

The main step of Bentov–Kumaresan's protocol is the fair reconstruction protocol (ii). By collecting all tokens, the parties can reconstruct the output value $f(x)$. However, malicious parties may abort so as to learn the output value while other parties do not. The fair reconstruction protocol prevents parties from aborting in the reconstruction phase. When malicious parties abort, they have to pay some amount of money for compensation to honest parties. It satisfies the following conditions:

(A) No honest party pays any penalty.

(B) If an adversary learns the reconstruction result, but an honest party cannot, then the honest party is compensated with coins. Furthermore, the compensation amounts are the same for any honest parties.

Note that honest parties are not guaranteed to receive compensation if an adversary aborts without learning the output value.

In summary, secure computation with penalties can be realized by executing a secure computation protocol for $\hat{f}$ and the fair reconstruction protocol. The next section shows Bentov–Kumaresan's fair reconstruction protocol in the $\mathcal{F}^*_{\mathrm{CR}}$-hybrid model.

3.2 Bentov–Kumaresan's Fair Reconstruction Protocol

Hereafter, we use T_i to denote Token_i. Suppose that each party P_i has a token T_i and a set of tags $(\mathsf{Tag}_1, \dots, \mathsf{Tag}_n)$ at the beginning of the fair reconstruction protocol. We assume that all parties agree on the penalty amount q, where honest parties are compensated with $\mathsf{coins}(q)$ when malicious parties abort with obtaining the output value. In the below, we successively explain a naïve approach, a solution for the two-party setting, and a solution for the n-party setting.

Naïve approach. Suppose that the number of parties is two. A naïve approach is to make a deposit transaction from P_1 to P_2 and a deposit transaction of the reverse direction as follows:

$$P_1 \xrightarrow[q,\tau]{T_2} P_2 \tag{1}$$

$$P_2 \xrightarrow[q,\tau]{T_1} P_1 \tag{2}$$

The above arrow means that "P_2 can receive $\mathsf{coins}(q)$ only if P_2 publishes the token T_2, otherwise $\mathsf{coins}(q)$ is refunded back to P_1" (see Section 2.3). The bottom arrow is similar. At first glance, it seems a fair reconstruction protocol satisfying conditions (A) and (B) in Section 3.1. However, it is not the case. For instance, when P_2 is malicious, P_2 can steal $\mathsf{coins}(q)$ from P_1 as follows: after establishing transaction (1), P_2 publishes the token T_2 without making transaction (2). As a result, honest P_1 loses $\mathsf{coins}(q)$. This violates condition (A).

Bentov-Kumaresan's solution. In order to avoid the above attack, Bentov–Kumaresan's fair reconstruction protocol for the two-party setting proceeds as follows:

$$P_1 \xrightarrow[q,\tau_2]{T_1 \wedge T_2} P_2 \tag{1}$$

$$P_2 \xrightarrow[q,\tau_1]{T_1} P_1 \tag{2}$$

where the rounds satisfy $\tau_1 < \tau_2$. (Hereafter, we assume $\tau_i < \tau_{i+1}$ for any integer i.) P_1 first makes a deposit transaction for $T_1 \wedge T_2$. Transaction (1) means that P_2 can receive $\mathsf{coins}(q)$ only if P_2 publishes both T_1 and T_2 in τ_2. Namely, it is necessary that both (Tag_1, T_1) and (Tag_2, T_2) are valid tag-token pairs to satisfy $\phi_{s,r}(T_1 \wedge T_2) = 1$. After making the first deposit transaction, P_2 makes a deposit transaction for T_1. Transaction (2) means that P_1 can receive $\mathsf{coins}(q)$ only if P_1 publishes T_1 in τ_1. In the claim phase, P_1 first publishes T_1, and then P_2 publishes both T_1 and T_2.

It is important to note that P_1 needs to make transaction (1) first. As a result, P_2 cannot claim this transaction without making transaction (2) since P_2 does not know T_1 yet. Also, the claims are performed in the reverse order of making the transactions, i.e., P_1 first claims.

If P_2 aborts after P_1 claims, then P_2 is penalized with $\mathsf{coins}(q)$ and P_1 is compensated with that coins. Thus, P_2 needs to publish T_2 in order not to lose $\mathsf{coins}(q)$. Also, both parties never are penalized if they behave honestly. Therefore, the above protocol satisfies the conditions (A) and (B) in Section 3.1.

We show Bentov-Kumaresan's solution for the n-party setting on the left side of Figure 1. As with the two-party case, parties make deposit transactions from the top and claim from the bottom in the n-party setting. Namely, the parties make transactions (1) to $(2n-2)$ and claim transactions $(2n-2)$ to (1).

Here, we describe an intuitive explanation that Bentov-Kumaresan's fair reconstruction protocol satisfies the condition (A) and (B). (See [5] for a formal security proof based on Definition 1.) It is trivial that no party loses coins if all parties behave honestly. Thus, we consider the case where there is a party to abort.

Let consider the case where an adversary aborts in the deposit phase. Since no honest party publishes his/her token, the adversary does not learn the reconstruction result nor receives any coins from honest parties. This case satisfies the condition (A) and (B).

Let consider the case where an adversary aborts in the claim phase. In order to learn the reconstruction result, the adversary must collude all parties that have not claimed yet to learn tokens that are not published. Every honest party holds $\mathsf{coins}(q)$ since he/she has already claimed and has got coins. This case also satisfies the condition (A) and (B).

Efficiency. Bentov–Kumaresan's fair reconstruction protocol requires n rounds for deposit phase and n rounds for claim phase, and thus it requires a total of $2n$ rounds. Also, it requires $2n-2$ calls of $\mathcal{F}^*_{\mathrm{CR}}$. Recall that the augmented function can be computed in a constant round for any function. Therefore, for any function, Bentov–Kumaresan's protocol for the secure computation with penalties can be SCC realized in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}^*_{\mathrm{CR}})$-hybrid model with $O(n)$ rounds and $O(n)$ calls of $\mathcal{F}^*_{\mathrm{CR}}$.

Table 2 Comparison of fair reconstruction protocols.

References	# of Rounds	# of Calling $\mathcal{F}^*_{\mathrm{CR}}$	Compensation Amount
Bentov–Kumaresan [4]	$2n$	$2n-2$	Equivalent
This work (Sect. 4)	8	$3n-4$	Non-equivalent

Algorithm 3 Secure computation with non-equivalent penalties $\mathcal{F}^*_{f,\mathrm{neq}}$.

```
Setup  The session identifier is $sid$. Running with parties $P_1, \ldots, P_n$, and an ideal adversary
  $\mathcal{S}$ that corrupts parties $\{P_i\}_{i \in C}$. Let $d$ be a parameter representing the safety deposit.
  Let $q$ denote the minimum penalty amount.
Input phase  Wait to receive the following messages.
  - $(\mathrm{input}, sid, ssid, i, x_i, \mathsf{coins}(d))$ from $P_i$ for all $i \in H$
  - $(\mathrm{input}, sid, ssid, \{x_i\}_{i \in C}, H', \mathsf{coins}(\sum_{i \in H'} q_i))$ from $\mathcal{S}$, where $H' \subseteq H$ and $q_i\ (\geq q)$ is
    the penalty amount for each $i \in H'$.
Output phase  Perform the following process.
  1) Send $(\mathrm{return}, sid, ssid, \mathsf{coins}(d))$ to each $P_r$ for $r \in H$.
  2) Compute $(y_1, \ldots, y_n) \leftarrow f(x_1, \ldots, x_n)$.
     - if $h' = 0$, then send message $(\mathrm{output}, sid, ssid, z_r)$ to $P_r$ for $r \in H$, and terminate.
     - If $0 < h' < h$, then send $(\mathrm{extra}, sid, ssid, \mathsf{coins}(q_i))$ to $P_i$ for each $i \in H'$, and
       terminate, where $h := |H|$.
     - If $h' = h$, then send message $(\mathrm{output}, sid, ssid, \{y_i\}_{i \in C})$ to $\mathcal{S}$.
  3) If $\mathcal{S}$ returns $(\mathrm{continue}, sid, ssid, H'')$, where $H'' \subseteq H$, then perform the following
     process:
       3-i) send $(\mathrm{output}, sid, ssid, y_i)$ to $P_i$ for all $i \in H$,
      3-ii) send $(\mathrm{payback}, sid, ssid, \mathsf{coins}(\sum_{i \in H'} q_i - \sum_{j \in H''} q_j)$ to $\mathcal{S}$ where $h'' = |H''|$,
     3-iii) send $(\mathrm{extrapay}, sid, ssid, \mathsf{coins}(q_i))$ to $P_i$ for each $i \in H''$.
  4) Else if $\mathcal{S}$ returns $(\mathrm{abort}, sid, ssid)$, send $(\mathrm{penalty}, sid, ssid, \mathsf{coins}(q_i))$ to $P_i$ for each
     $i \in H$.
```

4 Proposed Protocol

In this section, we introduce a special functionality called secure computation with non-equivalent penalties (Section 4.1). Then we design a protocol achieving this functionality in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}^*_{\mathrm{CR}})$-hybrid model. In particular, we design a new fair reconstruction protocol in the $\mathcal{F}^*_{\mathrm{CR}}$-hybrid model (Section 4.3), and putting it with a secure computation protocol for an augmented function into the $\mathcal{F}_{\mathrm{OT}}$-hybrid model as in Section 3.1. Notably, our protocol requires $O(1)$ rounds and $O(n)$ broadcasts only (See Table 1).

4.1 Secure Computation with Non-equivalent Penalties

In secure computation with penalties $\mathcal{F}^*_f$, all honest parties are compensated with the same amount of money $\mathsf{coins}(q)$. A new functionality, secure computation with non-equivalent penalties $\mathcal{F}^*_{f,\mathrm{neq}}$, is the same as $\mathcal{F}^*_f$ except that each honest party is compensated with $\mathsf{coins}(q)$ *or more*, i.e., the amount of compensation *may* be different with each party. For example, in $\mathcal{F}^*_{f,\mathrm{neq}}$, we allow the following situation: An honest P_1 is compensated with $\mathsf{coins}(q)$ but an honest P_2 is compensated with $\mathsf{coins}(2q)$.

See Algorithm 3 for a formal definition of $\mathcal{F}^*_{f,\mathrm{neq}}$. The difference with $\mathcal{F}^*_f$ is that a simulator can decide the amount q_i for each $i \in H'$ and inputs $\mathsf{coins}(\sum_{i \in H'} q_i)$ while a simulator in $\mathcal{F}^*_f$ must input $\mathsf{coins}(h'q)$ for $h' := |H'|$. We require that $q_i \geq q$ for all $i \in H'$, where q is the minimum amount of compensation.

We note that compensation happens only when a malicious party has stolen the output value. That is, $\mathcal{F}^*_f$ and $\mathcal{F}^*_{f,\mathrm{neq}}$ are the same if all parties behave honestly. By choosing q appropriately, it is possible to prevent malicious behavior, and then we obtain a protocol with fairness. In this sense, a new functionality $\mathcal{F}^*_{f,\mathrm{neq}}$ brings almost the same effect on $\mathcal{F}^*_f$.

4.2 Fair Reconstruction for Secure Computation with Non-equivalent Penalties

Following Bentov-Kumaresan's protocol, we construct a fair reconstruction protocol to realize secure computation with non-equivalent penalties. In order to realize secure computation with non-equivalent penalties, a fair reconstruction protocol needs to satisfy the following conditions:

(A) No honest party pays any penalty.

(B*) If an adversary learns the reconstruction result, but an honest party cannot, then the honest party is compensated with coins. Furthermore, the compensation is more than a predetermined amount.

Note that the difference between condition (B*) and condition (B) in Section 3.1 is the amount of compensations only. Namely, our fair reconstruction protocol does not guarantee that each honest party is compensated with the same amount of coins.

4.3 Our Fair Reconstruction Protocol

Our fair reconstruction protocol proceeds as follows (see also the right side of Figure 1):

Deposit phase

1) For $i \in \{1, \dots, n-1\}$, P_i makes a transaction to send P_n $\mathsf{coins}(q)$ with a circuit $\phi_{i,n}$ and a round number τ_4, where $\phi_{i,n}(x) = 1$ only if $x = T_1 \wedge \cdots \wedge T_n$.

2) P_n makes a transaction to send P_{n-1} $\mathsf{coins}((n-1)q)$ with a circuit $\phi_{n,n-1}$ and a round number τ_3, where $\phi_{n,n-1}(x) = 1$ only if $x = T_1 \wedge \cdots \wedge T_{n-1}$.

3) For $i \in \{1, \dots, n-2\}$, P_{n-1} makes a transaction to send P_i $\mathsf{coins}((n-1)q)$ with a circuit $\phi_{n-1,i}$ and a round number τ_2, where $\phi_{n-1,i}(x) = 1$ only if $x = T_{n-1} \wedge T_i$.

4) For $i \in \{1, \dots, n-2\}$, P_i makes a transaction to send P_{n-1} $\mathsf{coins}((n-2)q)$ with a circuit $\phi_{i,n-1}$ and a round number τ_1, where $\phi_{i,n-1}(x) = 1$ only if $x = T_{n-1}$.

Claim phase

5) P_{n-1} claims by publishing T_{n-1} in round τ_1 and receives $\mathsf{coins}((n-2)q)$ from each of $P_1, \dots, P_{n-2}$.

6) For $i \in \{1, \dots, n-2\}$, P_i claims by publishing $T_{n-1} \wedge T_i$ in round τ_2 and receives $\mathsf{coins}((n-1)q)$ from P_{n-1}.

7) P_{n-1} claims by publishing $T_1 \wedge \cdots \wedge T_{n-1}$ in round τ_3 and receives $\mathsf{coins}((n-1)q)$ from P_n.

8) P_n claims by publishing $T_1 \wedge \cdots \wedge T_n$ in round τ_4 and receives $\mathsf{coins}(q)$ from each of $P_1, \dots, P_{n-1}$.

Our fair reconstruction protocol requires eight rounds and $3n-4$ calls of $\mathcal{F}^*_{\mathrm{CR}}$. Since $\mathcal{F}_{\mathrm{OT}}$ is sufficient to compute any standard functionality in constant rounds, we can derive the following theorem. (We defer the proof to the full version.)

▶ **Theorem 2.** *Assuming the existing of one-way functions, for every n-party functionality f there exists a protocol that* SCC *realizes* $\mathcal{F}^*_{f,\mathrm{neq}}$ *in the* $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}^*_{\mathrm{CR}})$*-hybrid model. The protocol requires* $O(1)$ *rounds and* $O(n)$ *calls of* $\mathcal{F}^*_{\mathrm{CR}}$.

4.4 Idea behind Our Protocol

See Figure 2 that shows flows of the claim phase of Bentov–Kumaresan's fair reconstruction protocol and ours.[4] In Bentov–Kumaresan's protocol, parties publishes his/her token in serial order, i.e., each token is published in each round. (Token T_i is published in round τ_i.) Thus, their protocol requires $O(n)$ rounds.

On the other hand, our protocol enables to publish multiple tokens in one round to improve the round complexity. See step 6) in Section 4.3, the parties $P_1, \ldots, P_{n-2}$ publish their token in one round.

In the claim phase, our protocol proceeds as follows: We call $P_1, \ldots, P_{n-2}$ *middle parties* and P_{n-1} *aggregator*. In round τ_1, the aggregator P_{n-1} collects coins from all middle parties by publishing token T_{n-1}. After that, the middle parties publishes their tokens $T_1, \ldots, T_{n-2}$ and receive coins, which are more than they sent in round τ_1, from the aggregator P_{n-1} in round τ_2. In round τ_3, the aggregator P_{n-1} receives coins from P_n by publishing his/her token and all of middle parties' tokens. In the last round τ_4, P_n publishes the last token T_n and receives coins from every other party. As a result, all parties learn the reconstruction result and every party's wallet are balanced, i.e., it has neither loss nor gain.

We discuss the amount of coins sent in each transaction to satisfy the conditions (A) and (B*) below.

The amount of coins. In our protocol, P_n receives $\mathsf{coins}(q)$ from every other party in the last round τ_4. (See Figure 1.) In order to satisfy the condition (A), every wallet of $P_1, \ldots, P_{n-1}$ must hold $\mathsf{coins}(q)$ at the end of round τ_3. We show that our protocol satisfies this condition in Figure 3.

When we decide the amount of coins in rounds τ_1 and τ_2, we should note that the aggregator P_{n-1} cannot claim in round τ_3 if at least one of the middle parties abort in round τ_2. Since the aggregator sends more coins in round τ_2 than he/she received in round τ_1, his/her wallet holds negative amount of coins at the end of round τ_2. In order to satisfy the conditions (A) and (B*), it is necessary to satisfy that the aggregator's wallet holds positive amount of coins at the end of round τ_2 if at least one of the middle parties abort in round τ_2. The amounts of coins sent in rounds τ_1 and τ_2 are derived as follows.

Suppose that P_{n-1} gets $\mathsf{coins}(xq)$ from each of $P_1, \ldots, P_{n-2}$ in round τ_1, and each of $P_1, \ldots, P_{n-2}$ get $\mathsf{coins}((x+1)q)$ from P_{n-1} in round τ_2. In round τ_2, P_{n-1}'s wallet should have positive amount of coins unless all of $P_1, \ldots, P_{n-2}$ claims. Thus, we can derive x from the following equation: $(n-2)x > (n-3)(x+1)$. The least solution of the equation is $x = n-2$. Therefore, each middle party sends $\mathsf{coins}((n-2)q)$ to the aggregator in round τ_1 and the aggregator sends $\mathsf{coins}((n-1)q)$ to each middle party in round τ_2.

Security intuition. Let consider the case where one of the middle parties aborts in round τ_2. (See Figure 4.) Suppose that P_1 aborts in round τ_2, i.e., he/she does not publish T_1 and does not receive $\mathsf{coins}((n-1)q)$ from P_{n-1}. Note that P_1 must collude with P_n to learn the

[4] For ease of understanding, Figure 2 omits the transactions (among P_n and other parties) in the last round. (See transactions $(1), (2), \ldots, (n-1)$ in Figure 1.)

reconstruction result. Thus, the condition (B*) is satisfied since every wallet of $P_2, \dots, P_{n-1}$ holds $\mathsf{coins}(q)$ as the compensation at the end of the protocol. Furthermore, since no honest party does not pay a penalty, the condition (A) is satisfied. We can confirm that our protocol satisfies the conditions (A) and (B*) by the same way in the other cases.

▶ Remark 3. Compensations to honest parties may not be the same amount of coins. See P_{n-1} who receives $\mathsf{coins}((n-2)q)$ from each of $P_1, \dots, P_{n-2}$ in round τ_1. The amount of P_{n-1}'s compensation depends on the number of aborting parties in them. On the other hand, compensations for other parties are $\mathsf{coins}(q)$. Namely, P_{n-1} is the only party who can be compensated with more than $\mathsf{coins}(q)$.

▶ Remark 4. At first glance, it seems that rounds τ_3 and τ_4 need not be separated since P_n can already claim in τ_3. However, if these rounds are combined into one (i.e., $\tau_4 = \tau_3$), the modified protocol violates condition (A). Suppose all but P_n are malicious. First, in the deposit phase, the adversary makes the $n-1$ transactions to P_n honestly. However, after P_n makes the deposit transaction to P_{n-1}, the adversary waits for time to pass without making the subsequent transactions. Just before the end of τ_3, the adversary claims the transaction made by P_n and obtains $\mathsf{coins}((n-1)q)$. P_{n-1} can get that coins back by claiming $n-1$ transactions made by the adversary, however P_n may not claim due to the lack of time remaining. As a result, P_n may lose the coins, which violates the condition (A).

5 Conclusion

This paper focused on Bentov and Kumaresan's work [4] in secure computation with penalties. They showed that secure computation with penalties could be constructed with $O(n)$ rounds and $O(n)$ broadcasts for any function in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}^*_{\mathrm{CR}})$-hybrid model. Also, it was the open problem whether the round order could be improved to $O(1)$ with $O(n)$ broadcasts [13].

This paper presented a positive answer to this question in a relaxed setting. In Bentov-Kumaresan's protocol, every honest party can be compensated with the same amount of coins when an adversary aborts after learning the output value. On the other hand, in our setting, every honest party is guaranteed to be compensated with more than a predetermined amount of coins, but not the same amount. We formalized this new setting as secure computation with non-equivalent penalties. We showed that secure computation with non-equivalent penalties could be realized with $O(1)$ rounds and $O(n)$ broadcasts for arbitrary functions in the $(\mathcal{F}_{\mathrm{OT}}, \mathcal{F}^*_{\mathrm{CR}})$-hybrid model. In particular, we improved the fair reconstruction protocol [4], which is a key ingredient for realizing secure computation with penalties.

References

1 Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Łukasz Mazurek. Fair two-party computations via bitcoin deposits. In Rainer Böhme, Michael Brenner, Tyler Moore, and Matthew Smith, editors, *Financial Cryptography and Data Security*, pages 105–121, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

2 Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. Secure multiparty computations on bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 443–458, 2014. doi:10.1109/SP.2014.35.

3 Adam Back and Iddo Bentov. Note on fair coin toss via bitcoin. *CoRR*, abs/1402.3698, 2014. arXiv:1402.3698.

4 Iddo Bentov and Ranjit Kumaresan. How to use bitcoin to design fair protocols. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014*, pages 421–439, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

5 Iddo Bentov and Ranjit Kumaresan. How to use bitcoin to design fair protocols. Cryptology ePrint Archive, Report 2014/129, 2014.

6 Iddo Bentov, Ranjit Kumaresan, and Andrew Miller. Instantaneous decentralized poker. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 410–440, Cham, 2017. Springer International Publishing.

7 R Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, pages 364–369, New York, NY, USA, 1986. Association for Computing Machinery. doi:10.1145/12130.12168.

8 Bernardo David, Rafael Dowsley, and Mario Larangeira. Kaleidoscope: An efficient poker protocol with payment distribution and penalty enforcement. In Sarah Meiklejohn and Kazue Sako, editors, *Financial Cryptography and Data Security*, pages 500–519, Berlin, Heidelberg, 2018. Springer Berlin Heidelberg.

9 Juan A. Garay, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. Adaptively secure broadcast, revisited. In *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, PODC '11, pages 179–186, New York, NY, USA, 2011. Association for Computing Machinery. doi:10.1145/1993806.1993832.

10 Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer – efficiently. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, pages 572–591, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

11 Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas. Fair and robust multi-party computation using a global transaction ledger. In *Proceedings, Part II, of the 35th Annual International Conference on Advances in Cryptology – EUROCRYPT 2016 - Volume 9666*, pages 705–734, Berlin, Heidelberg, 2016. Springer-Verlag.

12 Joe Kilian. Founding crytpography on oblivious transfer. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 20–31, New York, NY, USA, 1988. Association for Computing Machinery. doi:10.1145/62212.62215.

13 Ranjit Kumaresan and Iddo Bentov. How to use bitcoin to incentivize correct computations. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 30–41, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2660267.2660380.

14 Ranjit Kumaresan and Iddo Bentov. Amortizing secure computation with penalties. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 418–429, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2976749.2978424.

15 Ranjit Kumaresan, Tal Moran, and Iddo Bentov. How to use bitcoin to play decentralized poker. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 195–206, New York, NY, USA, 2015. Association for Computing Machinery. doi:10.1145/2810103.2813712.

16 Ranjit Kumaresan, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. Improvements to secure computation with penalties. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 406–417, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2976749.2978421.

17 Andrew Y. Lindell. Legally-enforceable fairness in secure two-party computation. In Tal Malkin, editor, *Topics in Cryptology – CT-RSA 2008*, pages 121–137, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

18 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Cryptography Mailing list at https://metzdowd.com*, March 2009.

19 Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

20 Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, SFCS '86, pages 162–167, USA, 1986. IEEE Computer Society. doi:10.1109/SFCS.1986.25.

$$P_1 \xrightarrow[q,\tau_n]{T_1\wedge\cdots\wedge T_n} P_n \tag{1}$$

$$P_2 \xrightarrow[q,\tau_n]{T_1\wedge\cdots\wedge T_n} P_n \tag{2}$$

$$\vdots$$

$$P_{n-1} \xrightarrow[q,\tau_n]{T_1\wedge\cdots\wedge T_n} P_n \tag{$n-1$}$$

$$P_n \xrightarrow[(n-1)q,\tau_{n-1}]{T_1\wedge\cdots\wedge T_{n-1}} P_{n-1} \tag{n}$$

$$P_{n-1} \xrightarrow[(n-2)q,\tau_{n-2}]{T_1\wedge\cdots\wedge T_{n-2}} P_{n-2} \tag{$n+1$}$$

$$\vdots$$

$$P_4 \xrightarrow[3q,\tau_3]{T_1\wedge T_2\wedge T_3} P_3 \tag{$2n-4$}$$

$$P_3 \xrightarrow[2q,\tau_2]{T_1\wedge T_2} P_2 \tag{$2n-3$}$$

$$P_2 \xrightarrow[q,\tau_1]{T_1} P_1 \tag{$2n-2$}$$

$$P_1 \xrightarrow[q,\tau_4]{T_1\wedge\cdots\wedge T_n} P_n \tag{1}$$

$$P_2 \xrightarrow[q,\tau_4]{T_1\wedge\cdots\wedge T_n} P_n \tag{2}$$

$$\vdots$$

$$P_{n-1} \xrightarrow[q,\tau_4]{T_1\wedge\cdots\wedge T_n} P_n \tag{$n-1$}$$

$$P_n \xrightarrow[(n-1)q,\tau_3]{T_1\wedge\cdots\wedge T_{n-1}} P_{n-1} \tag{n}$$

$$P_{n-1} \xrightarrow[(n-1)q,\tau_2]{T_{n-1}\wedge T_{n-2}} P_{n-2} \tag{$n+1$}$$

$$P_{n-1} \xrightarrow[(n-1)q,\tau_2]{T_{n-1}\wedge T_{n-3}} P_{n-3} \tag{$n+2$}$$

$$\vdots$$

$$P_{n-1} \xrightarrow[(n-1)q,\tau_2]{T_{n-1}\wedge T_1} P_1 \tag{$2n-2$}$$

$$P_{n-2} \xrightarrow[(n-2)q,\tau_1]{T_{n-1}} P_{n-1} \tag{$2n-1$}$$

$$P_{n-3} \xrightarrow[(n-2)q,\tau_1]{T_{n-1}} P_{n-1} \tag{$2n$}$$

$$\vdots$$

$$P_1 \xrightarrow[(n-2)q,\tau_1]{T_{n-1}} P_{n-1} \tag{$3n-4$}$$

Figure 1 Bentov–Kumaresan's fair reconstruction protocol (left) and our fair reconstruction protocol (right) in the n-party setting: In the deposit phase, the transactions are created from top to bottom, i.e., (1) to $(2n-2)$ in the left protocol and (1) to $(3n-4)$ in the right protocol. In the claim phase, the transactions are claimed in the reverse direction, i.e., $(2n-2)$ to (1) in the left protocol and $(3n-4)$ to (1) in the right protocol. The horizontal lines separate each round. Namely, in the deposit (resp. claim) phase, transactions belonging to the same section are created (resp. claimed) in one round.

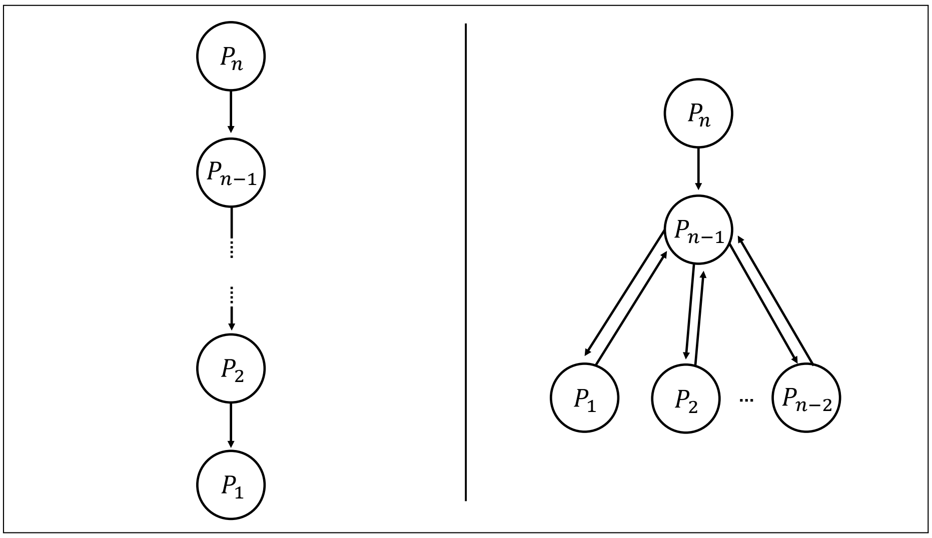

Figure 2 Flow of Bentov–Kumaresan's fair reconstruction protocol (left) and ours (right).

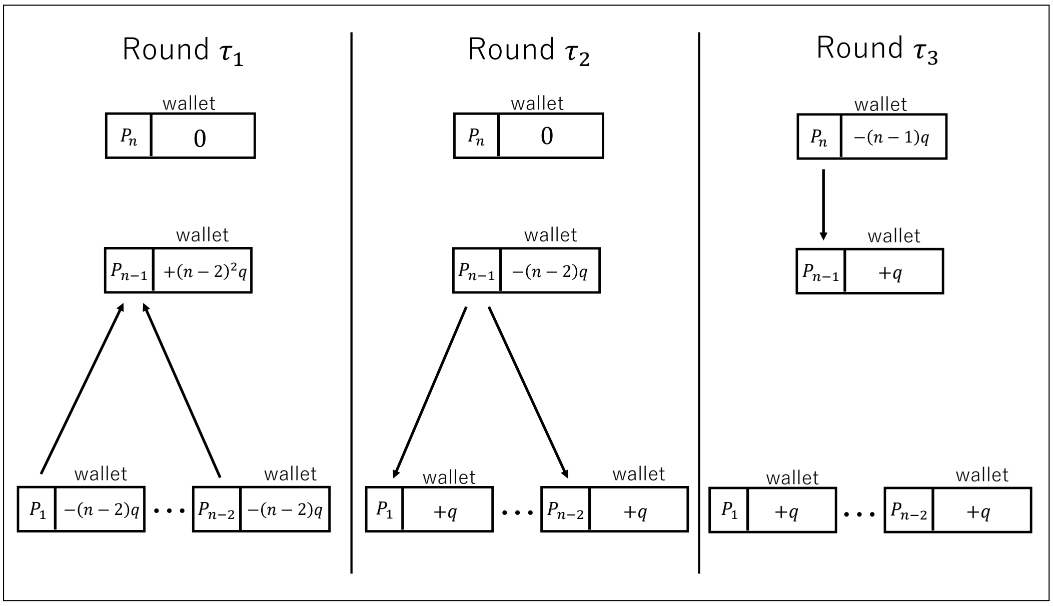

Figure 3 Coins flow in round τ_1 to τ_3 in the case where all parties behave honestly.

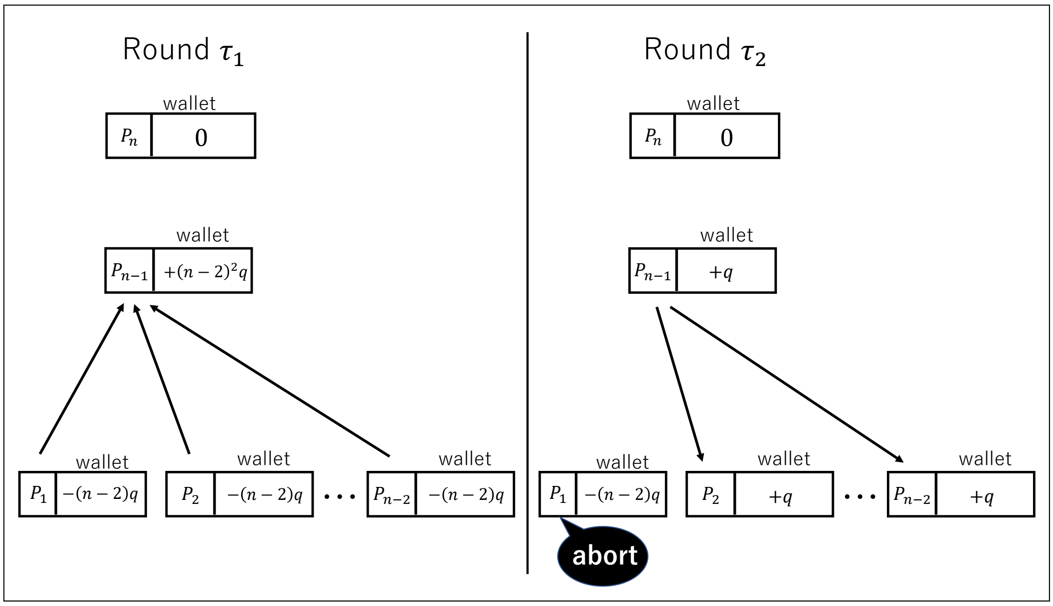

Figure 4 Coins flow in round τ_1 to τ_2 in the case where P_1 aborts.