Counterfeit Coin Problem Can Be Extended to Secure Multiparty Computation*

Shohei Kaneko[1], Yang Li[1], Kazuo Sakiyama[1], and Daiki Miyahara[1,2]

[1] The University of Electro-Communications, Tokyo, Japan
[2] National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

Abstract. The counterfeit coin problem is a well-known logic puzzle with many variants. In its classic form, the goal is to identify a single counterfeit (lighter) coin among eight visually identical coins using a balance scale with the minimum number of measurements. The answer is not three but two, achieved by cleverly dividing the eight coins. In this study, we extend this problem to the setting of secure multiparty computation. Specifically, given coins whose weights depend on private inputs, a balance scale can be used to compute a predetermined function over those inputs without revealing any additional information. We formalize secure computations with a balance scale and propose secure computation protocols for logical functions such as the logical AND and majority function. In particular, our proposed protocol can efficiently compute the majority function with only a single measurement, whereas existing protocols with everyday objects require several steps, such as card-based protocols (Eurocrypt'89), private PEZ protocols (TCC'19), and bag&ball-based protocols (CSF'21).

Keywords: Secure multiparty computations · Balance-scale · Coin · Card-based cryptography.

1 Introduction

The *counterfeit coin problem* is a well-known logic puzzle with many variants. The goal of the original one [7] is to identify a single counterfeit coin using a balance scale in the following setting: there are eight visually identical coins, among which only a single coin is lighter. For this, we are allowed to use a balance scale, but the number of the measurements must be minimum.

Clearly, three measurements are enough to identify the lighter coin because $8 \leq 2^3$. However, the minimum number is two; the key is to first place *three*

* This paper appears in Proceedings of FPS 2025. This version of the contribution has been accepted for publication, after peer review but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: https://doi.org/10.1007/978-3-032-20026-6_2. Use of this Accepted Version is subject to the publisher's Accepted Manuscript terms of use: https://www.springernature.com/gp/open-research/policies/accepted-manuscript-terms.

coins on each of the two plates among the eight coins. Then, we have two cases as follows:

- If the balance tilts, then the desired coin is among the three coins on the plate tilting up. In a similar manner, we compare the weights of any two coins among the three coins. If the balance tilts, then the desired coin is the one on the plate tilted up; otherwise, it is the one not on the plates.
- If the balance does not tilt, then the desired coin is among the two coins not on the plates. Finally, we compare the weights of two coins. The desired coin is the one on the plate tilting up.

That is, information about a balance being static implies not only the weights of coins on the balance, but also those of coins not placed on the balance.

In this study, we extend this famous problem to the setting of *secure multi-party computation*. That is, given coins whose weights depend on private inputs, a balance scale enables us to compute a predetermined function over those inputs without revealing any information beyond what can be inferred from the output. We initiate this line of research by developing *balance-based cryptographic protocols*.

1.1 Contributions

In this study, we develop balance-based cryptography and propose balance-based protocols for Boolean functions, such as the logical AND, the majority function, and more general threshold functions. We prove the correctness and security of our protocols within a computational model that we formalize. Specifically, we present a rigorous representation for a balance and define the actions over it. A key strength of our protocols is that only a single measurement suffices to compute various functions. Thus, our model opens up not only a new direction of research, such as developing a zero-knowledge proof protocol for the solution of Sudoku using an everyday object (cf., [8,26]), but also practical demonstrations of didactic value for non-experts, including high-school students (cf., [9,16,24]).

This study is inspired by the counterfeit coin problem [7] introduced earlier. Our observation is that the use of a balance scale can be viewed as performing a form of computation to identify the counterfeit coin. In our protocols, we represent a bit value by the weight of a coin. That is, values 0 and 1 are represented by coins with weights w and w' such that $w < w'$, respectively. This encoding enables simple bit addition by merely stacking the corresponding coins together, without the need to build an adder circuit. Furthermore, since a balance scale can compare weights, it naturally facilitates the computation of symmetric Boolean functions, where the output depends only on the sum of the input bits.

1.2 Concurrent and Independent Work

Similar balance-based protocols have been proposed concurrently and independently by Ruangwises [25]. He introduced a balance scale and coins as a new

Table 1: Comparison to the concurrent and independent protocols [25], where n is the number of inputs.

Function	Protocol	#Coins	#Comparisons	Note
AND	[25]	$3n$	1	
	Ours	$3n$	1	
Threshold	[25]	$2n$	1	Uses custom weight
	Ours	$6n$	1	
Majority	Ours	$2n$	1	

alternative to a deck of cards in card-based cryptography [5, 20], and proposed secure computation protocols for any Boolean function. Four types of protocols were constructed to compute the AND function, threshold functions, symmetric functions, and any Boolean functions. However, it should be noted that some of his proposed protocols require special items:

- His threshold-function protocol requires custom weight, say $\frac{w'-w}{2}$.
- His symmetric-function protocol requires a pen for marking and bags for holding coins.
- His protocol for any Boolean function requires bags.

Ruangwises' method shares the same foundation as our protocols, namely the use of a balance scale and two types of indistinguishable coins (i.e., heavy and light), but our protocols require no such additional items. In addition, while Ruangwises' work was inspired by our earlier research [10], the present work is directly inspired by the classical counterfeit coin problem [7], which we reinterpret in the framework of secure computation using a balance scale. Our key contribution is to establish a formal computational model that links this classical problem to secure computation, providing a new foundation for future protocols[3]. Our current results do not yet cover all Boolean functions, but we present an efficient protocol for the majority (threshold) function that requires no special weights and operates solely with two types of coins. A detailed summary can be found in Table 1.

1.3 Related Work

Balance Scale. Balance scales have been employed from ancient civilizations to the present day, serving both practical and symbolic roles across diverse cultures and disciplines. Historically, they were essential tools in commerce for ensuring fair trade. In educational settings, balance scales facilitate the teaching of fundamental scientific principles like mass, leverage, and equilibrium. Additionally,

[3] Although an earlier version of this paper was presented domestically at the non-refereed symposium DICOMO 2023 (9A-2), our independent studies on balance-based zero-knowledge proof protocols for pencil-and-paper puzzles [10] and on balance-based voting protocols for multiple candidates [11] were published earlier.

they have become enduring motifs representing fairness and impartiality, exemplified by the depiction of the goddess Themis holding a scale in judicial iconography worldwide. Balance scales not only remain integral physical instruments but also powerful symbols embodying the universal human pursuit of balance, fairness, and precision.

Information-Theoretic Perspective. The counterfeit coin problem is classically studied in information theory to determine the lower bound on the number of measurements required to identify a target [3,6]. These works primarily focus on the search complexity to minimize the queries needed to resolve the uncertainty of the system (i.e., finding the counterfeit coin). In contrast, our work views a balance scale from a computational perspective: we treat the measurement mechanism as a physical realization of a threshold gate. Therefore, unlike the classical search problems, our protocols are designed to securely evaluate specific Boolean functions rather than to optimize search strategies.

Objects and Cryptography. Research on cryptographic protocols based on everyday physical objects has been gaining significant attention, such as a deck of playing cards [5,8], a PEZ dispenser [1,2,23], coins [15], and balls in bags [17]. In this line of research, reducing the numbers of objects and steps is a hot topic and is closely related to combinatorics. A primary advantage of this approach is its ability to educate the general public about security without requiring specialized knowledge. By presenting cryptographic techniques through familiar and easily understandable objects, these protocols allow a broader audience to grasp the fundamental concepts of secure computations. This accessibility not only increases public interest in cryptographic technologies but also encourages wider participation in research and development, contributing to the continuous advancement of cryptographic methods.

Card-Based Cryptography. Prior to the 2010s, research on card-based cryptography [4,20,22,28] focused primarily on proposing card-based protocols for securely computing specific logical functions. Since Mizuki and Shizuya [21] constructed a computational model to formally describe card-based protocols in 2014, it has become possible to prove correctness and security within a formal model. Based on this computational model, Koch et al. [14] devised a diagram method to visually represent the state transitions of a protocol in 2015, and it has now become a standard method for proving correctness and security. It should be noted that a lower bound on the number of cards required was also determined in the computational model (cf. [12, 13]). The emergence of computational models in the field of card-based cryptography has led to several important studies.

1.4 Outline

This paper is organized as follows. In Sect. 2, we present AND protocols using a balance scale as an example. In Sect. 3, we formalize balance-based protocols. In Sects. 4 and 5, we construct a balance-based cryptographic protocol for the

threshold function and majority function, respectively, and prove their correctness and security. In Sect. 6, we discuss our computational model constructed in this paper and future challenges. Section 7 concludes this paper.

2 Examples of Balance-Based Secure Computation

In this section, we provide illustrative examples of secure computation using a balance scale. We first show the simplest non-trivial case of a two-input AND protocol, and subsequently generalize it to the multiple-input setting.

2.1 Two-Input AND Protocol

Assume that Alice and Bob have private inputs $x_1, x_2 \in \{0, 1\}$, respectively, and are willing to obtain $x_1 \wedge x_2$. They have a balance scale and two visually identical coins of weight w, w' such that $w < w'$. The protocol proceeds as follows.

1. Alice (resp. Bob) places the w coin on the left plate if $x_1 = 0$ (resp. $x_2 = 0$); otherwise, places the w' coin. Note that each player should place a coin in such a way that neither knows which coin the other player has placed.
2. Place two additional w' coins on the right plate.
3. Obtain the value of $x_1 \wedge x_2$ from the balance as follows.
   - If the balance tilts to the right, then $x_1 \wedge x_2 = 0$.
   - If it is balanced, then $x_1 \wedge x_2 = 1$.

Clearly, this protocol can compute the two-input AND function; if and only if $x_1 \wedge x_2 = 1$, the two coins placed on the left side weigh $2w'$. It should be noted that when the two coins on the left weigh either $2w$ (i.e., $x_1 + x_2 = 0$) or $w + w'$ (i.e., $x_1 + x_2 = 1$), the balance must tilt in the same way. We therefore assume an *ideal* balance that tilts at a constant speed regardless of the weight difference.

2.2 Multiple-Input AND Protocol

We extend the previous protocol to an n-input AND protocol for $n \geq 2$. This extension is natural, and the protocol proceeds as follows.

1. Each player holding $x_i \in \{0, 1\}$ places a w coin on the left plate if $x_i = 0$; otherwise, places a w' coin.
2. Place n w' coins on the right plate.
3. The output $x_1 \wedge x_2 \wedge \cdots \wedge x_n$ is obtained from the balance as follows.
   - If the balance tilts to the right, then $x_1 \wedge \cdots \wedge x_n = 0$.
   - If it is balanced, then $x_1 \wedge \cdots \wedge x_n = 1$.

In this protocol, each of the n players has two coins (w and w'), and n additional w' coins are placed. The total number of coins used in this protocol is $3n$. Clearly, the balance is used only once.

3 Formalizing Balance-Based Protocols

In this section, we present a formal treatment of balance-based cryptography by developing a computational model for balance-based secure computation. Here, we follow a computational model for card-based cryptography [14, 19, 21, 27].

3.1 Notation

In the protocol mentioned above, one balance scale, two coins of weight w and four coins weight w' are used. We assume an *ideal* balance scale that always tilts to the heavier side, with the tilt looking the same regardless of the magnitude of the weight difference.

Let $\mathcal{W}$ be a finite multiset of positive real numbers, i.e., $\mathcal{W} \subset \mathbb{R}^+$ with multiplicities. For $w \in \mathcal{W}$, we denote by $c(w)$ a *coin* of weight w, which can be placed on the balance. Although coins may have different weights, they are visually indistinguishable. A sequence of d ordered coins $\Gamma = (c(w_1), c(w_2), \ldots, c(w_d))$ is called a *d-coin sequence* from $\mathcal{W}$ if $\{w_1, w_2, \ldots, w_d\} = \mathcal{W}$. Let $\mathsf{Seq}^{\mathcal{W}}$ denote a set of all coin sequences from $\mathcal{W}$:

$$\mathsf{Seq}^{\mathcal{W}} \coloneqq \{\Gamma \mid \Gamma \text{ is a coin sequence from } \mathcal{W}\}.$$

For a coin sequence $\Gamma = (c(w_1), c(w_2), \ldots, c(w_d)) \in \mathsf{Seq}^{\mathcal{W}}$, we define the following actions.

- (comp, L, R), for disjoint index subsets $L, R \subset \{1, 2, \ldots, d\}$, compares the coins at the positions specified by L and R and returns an observation in $\{L > R, L < R, L = R\}$ corresponding to the tilt of the balance:

$$\mathsf{comp}_{L,R}(\Gamma) \to \begin{cases} L > R, & \text{if } \sum_{i \in L} w_i > \sum_{j \in R} w_j, \\ L < R, & \text{if } \sum_{i \in L} w_i < \sum_{j \in R} w_j, \\ L = R, & \text{otherwise.} \end{cases}$$

- (shuf, Π), for a set of permutations $\Pi \subseteq S_d$ (where S_d denotes the symmetric group of degree d), rearranges Γ according to a permutation $\pi \in \Pi$, chosen uniformly at random:

$$\mathsf{shuf}_{\Pi}(\Gamma) \coloneqq \pi(\Gamma).$$

  Since coins are visually indistinguishable, any shuffle results in a sequence that appears identical, and thus the observation of this action is defined as $\perp$.
- (result, y), for the output value $y \in \{0, 1\}$, occurs only once when the protocol terminates and specifies the protocol's output as y.

3.2 Protocol Definition

Building on the above actions, we formally define a balance-based protocol.

Algorithm 1 Two-input AND protocol $(\mathcal{W}, U = \{\Gamma^{00}, \Gamma^{01}, \Gamma^{10}, \Gamma^{11}\}, Q, A)$.

Require: $\mathcal{W} = \{w, w, w', w', w', w'\}$,
Ensure: Logical AND value of the two input bits
1: **procedure** TWO-INPUT AND
2: **if** $(\mathsf{comp}, \{1,3\}, \{5,6\}) \to L = R$ **then**
3: $(\mathsf{result}, 1)$
4: **else if** $(\mathsf{comp}, \{1,3\}, \{5,6\}) \to L < R$ **then**
5: $(\mathsf{result}, 0)$
6: **end if**
7: **end procedure**

Definition 1. *A balance-based protocol is a 4-tuple $\mathcal{P} = (\mathcal{W}, U, Q, A)$ defined as follows:*

- *$\mathcal{W}$ is a multiset of weights.*
- *$U \subseteq \mathsf{Seq}^{\mathcal{W}}$ is a set of input coin sequences.*
- *Q is a finite state set that contains the initial state q_0 and the final state q_f.*
- *$A\colon Q \times \mathsf{Obs} \to Q \times \mathsf{Action}$ is an action function that determines the next state and action from the current state and the observation of the current action. Here, $\mathsf{Obs} \coloneqq \{L > R, L < R, L = R\} \cup \{\bot\}$ denotes the set of observable results of actions, and Action is the set of actions introduced above.*

Example. The two-input AND protocol introduced in Sect. 2.1 is formally denoted by

$$(\{w, w, w', w', w', w'\}, \{\Gamma^{00}, \Gamma^{01}, \Gamma^{10}, \Gamma^{11}\}, \{q_0, q_1, q_\mathrm{f}\}, A)$$

where

$$\begin{aligned}
\Gamma^{00} &= (c(w), c(w'), c(w), c(w'), c(w'), c(w')),\\
\Gamma^{01} &= (c(w), c(w'), c(w'), c(w), c(w'), c(w')),\\
\Gamma^{10} &= (c(w'), c(w), c(w), c(w'), c(w'), c(w')),\\
\Gamma^{11} &= (c(w'), c(w), c(w'), c(w), c(w'), c(w')),
\end{aligned}$$

and the action function A is as follows:

1. $A(q_0, \bot) = (q_1, (\mathsf{comp}, \{1,3\}, \{5,6\}))$;
2. $A(q_1, L = R) = (q_\mathrm{f}, (\mathsf{result}, 1))$;
3. $A(q_1, L < R) = (q_\mathrm{f}, (\mathsf{result}, 0))$.

A pseudocode for the two-input AND protocol is shown in Algorithm 1.

3.3 Correctness and Security

In this section, we formally define the security and correctness of a protocol $\mathcal{P}$. For an input $x \in \{0,1\}^n$, the execution of $\mathcal{P}$ yields a sequence of observations, each element of which belongs to Obs. We call this sequence an *observation trace for* x and denote it by $\mathsf{obs}(x) \in \mathsf{Obs}^*$.

Definition 2 (Correctness). *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function. A protocol $\mathcal{P} = (\mathcal{W}, U, Q, A)$ computes f if it satisfies the following two conditions:*

1. *Each input coin sequence $\Gamma^x \in U$ is in one-to-one correspondence with the input $x \in \{0,1\}^n$, according to the encoding rule specified by the protocol.*
2. *For every input $x \in \{0,1\}^n$, the output of $\mathcal{P}$ based on $\mathsf{obs}(x)$ equals $f(x)$ with probability 1.*

For the *security* of $\mathcal{P}$, we consider a semi-honest model. It means that no information other than the output can be obtained from $\mathsf{obs}(x)$ for any input x.

Definition 3 (Security). *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function. A protocol $\mathcal{P}$ computing f is* secure *if, for any two inputs $x, x' \in \{0,1\}^n$ with $f(x) = f(x')$, it holds that $\mathsf{obs}(x)$ and $\mathsf{obs}(x')$ are identically distributed.*

3.4 Diagram

In order to visually verify the correctness and security of the protocol, this paper presents a diagram representing the state transitions of a protocol. This diagram method was originally proposed by Koch et al. [14] to visually show the state transitions of a card-based protocol, and an extended version introducing *probability traces* has been proposed for card-based protocols by Mizuki and Komano [19]. Inspired by these studies, we depict such a diagram for balance-based secure computation, using the two-input AND protocol as an example.

Figure 1 depicts a diagram for the two-input AND protocol $\mathcal{P} = (\mathcal{W}, U, Q, A)$. Each "box" in the figure corresponds to a *state* that shows a probability distribution over $\mathsf{Seq}^{\mathcal{W}}$, given a prefix of the observation trace. The transitions are as follows:

- The topmost state contains each coin sequence $\Gamma \in U$, i.e., input coin sequences. Let $\Gamma^x \in U$ be the coin sequence corresponding to the input $x \in \{0,1\}^2$. At the beginning of $\mathcal{P}$, the coin sequence placed on the table is exactly Γ^x if and only if the input is x. Let p_x denote the symbolic variable representing this probability. Then the topmost state is the mapping $\Gamma^x \mapsto p_x$, and $\Gamma \mapsto 0$ for all $\Gamma \notin U$.
- The topmost state branches via the edge labeled with the comparison action and the resulting observation. It transitions to the right state if and only if $x = 11$, i.e., with probability p_{11}. Once we observe $L = R$, the posterior distribution is concentrated on Γ^{11}, i.e., $\Gamma^{11} \mapsto 1$.
- Otherwise, it transitions to the left state with probability $p_{00} + p_{01} + p_{10}$. In this case, we have $\Gamma^x \mapsto \frac{p_x}{p_{00}+p_{01}+p_{10}}$ for each $x \in \{00, 01, 10\}$.
- This protocol satisfies correctness because the result action outputs $f(x)$ for every input $x \in \{0,1\}^2$. Security also holds because for any input $x \in \{00, 01, 10\}$, the observation is always $L < R$.

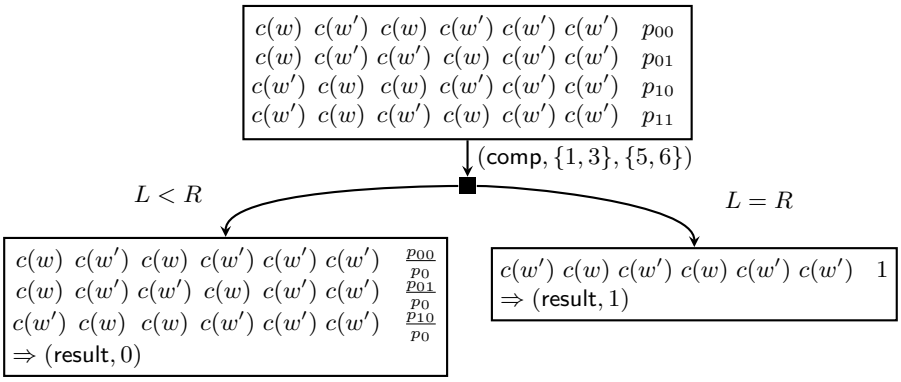

Fig. 1: Diagram of our two-input AND protocol, where $p_0 = p_{00} + p_{01} + p_{10}$

4 Threshold Function

In this section, we propose a balance-based protocol for computing threshold functions and give its formal description. A function $f_{(t,n)} \colon \{0,1\}^n \to \{0,1\}$ is called a (t,n)-threshold function if it is defined as

$$f_{(t,n)}(x_1, x_2, \ldots, x_n) = \begin{cases} 0, & \text{if } \sum_{i=1}^{n} x_i < t, \\ 1, & \text{otherwise.} \end{cases}$$

That is, $f_{(t,n)}$ outputs 1 if at least t of the n input bits are 1, and 0 otherwise. The n-input AND function corresponds to the (n,n)-threshold function.

4.1 Idea

To compute $f_{(t,n)}$, we extend the n-input AND protocol described in Sect. 2.2. In that protocol, the n coins placed on the right plate (i.e., n $c(w')$s) serve as a threshold, allowing one to determine whether the total weight of the n coins placed by the n players on the left plate is less than or equal to nw'. To generalize this to threshold functions, we replace the n $c(w')$s with a single coin $c(w'')$ such that

$$(t-1)w' + (n-t+1)w < w'' < tw' + (n-t)w.$$

This enables the balance to distinguish whether the number of $c(w')$s, i.e., input bits equal to 1, reaches t, but it requires introducing an additional coin of weight w'', which cannot be represented by using only $c(w)$ and $c(w')$.[4]

Our (t,n)-threshold function protocol overcomes this problem by simply doubling the number of coins. Specifically, each of the n players places two coins on the left plate: two $c(w')$s if the input bit is 1, and two $c(w)$s otherwise. As a result, the total weight on the left plate increases in steps of $2w$ or $2w'$, and the

[4] This method is essentially the same as the existing one [25].

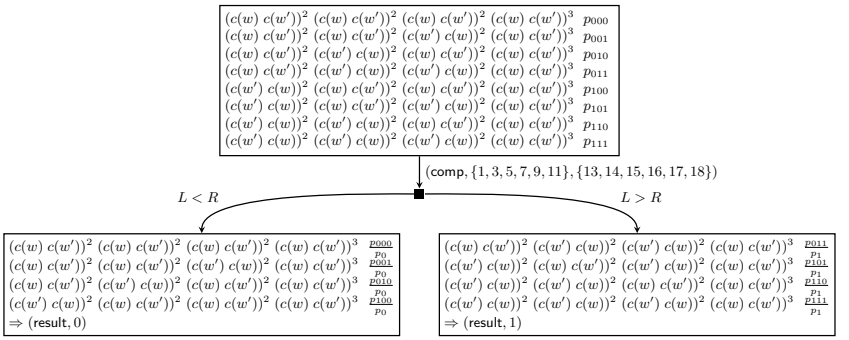

Fig. 2: Diagram of the (2,3)-threshold function protocol to compute $f_{(2,3)}$, where $p_0 = p_{000} + p_{001} + p_{010} + p_{100}$ and $p_1 = p_{011} + p_{101} + p_{110} + p_{111}$. Here, $(c(w))^\ell$ denotes ℓ $c(w)$s, and $(c(w)\ c(w'))^\ell$ represents ℓ pairs consisting of one $c(w)$ and one $c(w')$.

threshold can now be represented exactly by using only $c(w)$ and $c(w')$. That is, it suffices to place $(2n - 2t + 1)$ of $c(w)$ and $(2t - 1)$ of $c(w')$ on the right plate, because it holds that

$$2(t-1)w' + 2(n-t+1)w < 2nw + (w'-w)(2t-1) < 2tw' + 2(n-t)w.$$

Consequently, $f_{(t,n)}$ can be computed without introducing any additional coin type.

4.2 Procedure

Assume that each of n players have private input $x_i \in \{0, 1\}$, and are willing to obtain $f_{(t,n)}(x_1, x_2, \ldots, x_n)$. The protocol proceeds as follows.

1. Each player holding $x_i \in \{0, 1\}$ places two $c(w)$s on the left plate if $x_i = 0$; otherwise, places two $c(w')$s.
2. Place $(2n - 2t + 1)$ $c(w)$s and $(2t - 1)$ $c(w')$s on the right plate.
3. The output value is obtained from the balance as follows.
   - If the balance tilts to the left, then output 0.
   - If the balance tilts to the right, then output 1.

In this protocol, each of the n players has four coins (two $c(w)$s and two $c(w')$s), and $2n$ additional coins are used. The total number of coins used in this protocol is $6n$. Clearly, the balance is used only once. This protocol satisfies correctness and security clearly from the above description.

A pseudocode of the protocol for computing the (t, n)-threshold function is presented in Algorithm 2. Here, for a coin sequence Γ and a bit string x, $\Gamma[i]$ and $x[i]$ denote the i-th element, with indices starting from 1. As a concrete example, a diagram for the case $(t, n) = (2, 3)$ is shown in Fig. 2.

Algorithm 2 (t, n)-threshold function protocol $(\mathcal{W}, U, Q, A)$.

Require: $\mathcal{W} = \{w, w, \ldots, w, w', w', \ldots, w'\}$
Require: each input $x \in \{0,1\}^n$ and a coin sequence $\Gamma^x \in U$
Require: $\Gamma^x[4i-3] = \Gamma^x[4i-1] = c(w)$ and $\Gamma^x[4i-2] = \Gamma^x[4i] = c(w')$ if $x[i] = 0$ for $1 \le i \le n$
Require: $\Gamma^x[4i-3] = \Gamma^x[4i-1] = c(w')$ and $\Gamma^x[4i-2] = \Gamma^x[4i] = c(w)$ if $x[i] = 1$ for $1 \le i \le n$
Ensure: (t, n)-threshold function
1: **procedure** (t, n)-THRESHOLD
2: **if** $(\mathsf{comp}, \{2k-1 \mid 1 \le k \le 2n\}, \{k \mid 4n+1 \le k \le 6n\}) \to L > R$ **then**
3: $(\mathsf{result}, 1)$
4: **else if** $(\mathsf{comp}, \{2k-1 \mid 1 \le k \le 2n\}, \{k \mid 4n+1 \le k \le 6n\}) \to L < R$ **then**
5: $(\mathsf{result}, 0)$
6: **end if**
7: **end procedure**

5 Majority Voting Function

In this section, we propose a majority voting protocol that uses fewer coins than our threshold-function protocol. A majority function $\mathsf{maj}\colon \{0,1\}^n \to \{0,1\}$ can be defined as

$$\mathsf{maj}(x_1, x_2, \ldots, x_n) = \begin{cases} 0, & \text{if } \sum_{k=1}^{n} x_k \le \frac{n}{2}, \\ 1, & \text{otherwise.} \end{cases}$$

In other words, maj outputs 0 if at least half of the n input bits are 0, and 1 otherwise.

5.1 Idea

The main idea is to adapt the threshold-function protocol without doubling the number of coins presented in Sect. 4.1. Recall that each player holds two coins, $c(w)$ and $c(w')$, and places one of them on the balance according to their input. Thus, the number of $c(w')$ placed corresponds to the number of 1s in the input bits, while the number of $c(w')$ *not* placed corresponds to the number of 0s, i.e., it can be regarded as the "complement" of the placed coins. By comparing the weights of the placed and unplaced coins, one can determine whether the number of 1s exceeds $\frac{n}{2}$ or not.

5.2 Procedure.

Assume that each of n players have private input $x_i \in \{0,1\}$, and are willing to obtain $\mathsf{maj}(x_1, x_2, \ldots, x_n)$. The protocol proceeds as follows.

1. Each of the n players places two coins depending on their input:
   - If the input is 0, place $c(w')$ on the left plate and $c(w)$ on the right.
   - If the input is 1, place $c(w)$ on the left plate and $c(w')$ on the right.

Algorithm 3 Majority voting protocol $(\mathcal{W}, U, Q, A)$.

Require: $\mathcal{W} = \{w, w, \ldots, w, w', w', \ldots, w'\}$
Require: each input $x \in \{0,1\}^n$ and a coin sequence $\Gamma^x \in U$
Require: $\Gamma^x[2i-1] = c(w)$ and $\Gamma^x[2i] = c(w')$ if $x[i] = 0$ for $1 \leq i \leq n$
Require: $\Gamma^x[2i-1] = c(w')$ and $\Gamma^x[2i] = c(w)$ if $x[i] = 1$ for $1 \leq i \leq n$
Ensure: maj of the n input bits
1: **procedure** MAJORITY VOTING
2: **if** $(\mathsf{comp}, \{2k-1 \mid 1 \leq k \leq n\}, \{2k \mid 1 \leq k \leq n\}) \to L < R$ **then**
3: $(\mathsf{result}, 0)$
4: **else if** $(\mathsf{comp}, \{2k-1 \mid 1 \leq k \leq n\}, \{2k \mid 1 \leq k \leq n\}) \to L > R$ **then**
5: $(\mathsf{result}, 1)$
6: **end if**
7: **end procedure**

2. The output value is obtained from the balance as follows.
   – If the balance tilts to the left, then output 0.
   – If the balance tilts to the right, then output 1.

This protocol obviously satisfies correctness and security. The total number of coins is $2n$. A pseudocode of the protocol is presented in Algorithm 3.

5.3 Diagram

As a concrete example, Fig. 3 shows a diagram for the three-input majority voting protocol. In this protocol, six coins are given as input. The coins in odd positions are compared with those in even positions. If $L < R$ is observed, then the input is one of the values in $\{000, 001, 010, 100\}$, which clearly satisfies correctness. If $L > R$ is observed, then the input is one of the values in $\{011, 101, 110, 111\}$. Information obtained from the observation does not reveal anything beyond the output value.

6 Discussion

In this section, we discuss the power of our computational power constructed in Sect. 3.2. We then discuss the efficiency of our AND protocol, threshold-function protocol, and majority voting protocol presented in Sects. 2.1, 4 and 5.

6.1 Power of Our Computational Model

We show that our computational model also captures the existing protocols presented by Ruangwises [25], which require additional items such as a pen and bags. First, his threshold-function protocol using custom weight can be formalized simply by including the custom weight into a set of weights. We therefore focus on his protocols using a pen and bags.

As an example, we introduce his protocol for computing the three-input XOR function, which proceeds as follows.

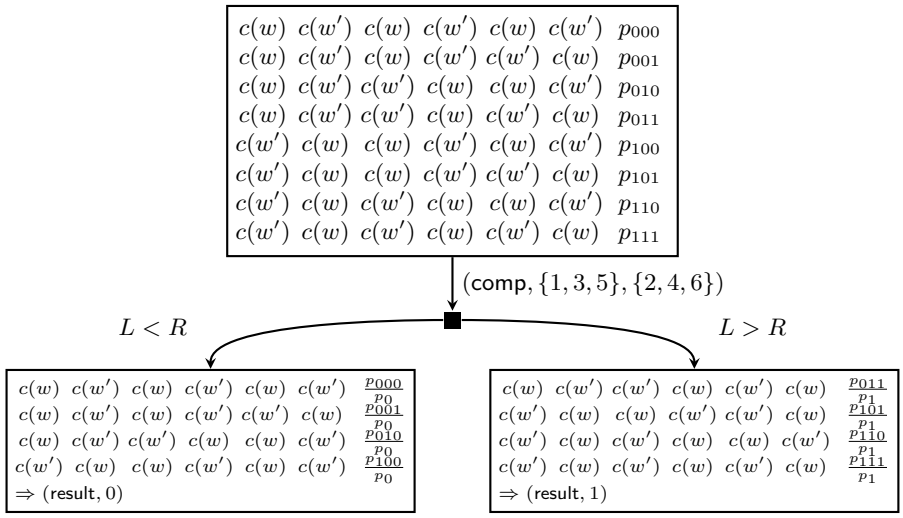

Fig. 3: Diagram of the three-input majority voting protocol, where $p_0 = p_{000} + p_{001} + p_{010} + p_{100}$ and $p_1 = p_{011} + p_{101} + p_{110} + p_{111}$.

1. Prepare two bags B_1 and B_3, where each B_i contains i $c(w')$ and $3-i$ $c(w)$, corresponding to the case that the sum of the input bits equals i.
2. Shuffle the two bags.
3. Prepare a special bag B_{s}, in which each player put $c(w')$ if its input bit is 1 and put $c(w)$ otherwise. Additionally, make a mark inside B_{s} using the pen. We note that B_{s} represents the sum of the actual inputs.
4. Pick one of the "non-special" bags and shuffle it together with B_{s}.
5. Compare the two bags. If their weights are equal, then output 1 and terminate because it means that the sum of the inputs is either 1 or 3; otherwise, if no other non-special bag remains, then output 0 and terminate.
6. Shuffle the two bags again.
7. Open both bags to find the mark and identify B_{s}. Discard the other bag and return to step 4.

Here, each bag collects several coins into a single unit so that they can be shuffled together as one object, rather than shuffling individual coins. Recall that our model defines a shuffle as selecting uniformly at random from a given set of permutations. Therefore, a shuffle involving bags can also be represented within our model. Interestingly, our model can also represent the operation of making a mark inside a bag. Let $c(w')$ and $c(w)$ correspond to the marked and unmarked bags, respectively. Then detecting the mark is equivalent to comparing $c(w')$ and $c(w)$: the bag corresponding to the heavier coin is identified as the marked bag.

The protocol introduced above is formalized step by step, following the steps described above.

1. We are given a 15-coin sequence $\Gamma = (c_1, c_2, \ldots, c_{15})$ such that:

- the four coins c_1, c_2, c_4, and c_6 correspond to B_{s}, and $c_1 = c(w')$;
- the four consecutive coins from c_8 (resp. c_{12}) correspond to B_1 (resp. B_3). That is, we have

$$c_8 = c(w), \quad c_9 = c(w'), \quad c_{10} = c(w), \quad c_{11} = c(w),$$
$$c_{12} = c(w), \quad c_{13} = c(w'), \quad c_{14} = c(w'), \quad c_{15} = c(w').$$

2. Apply $(\mathsf{shuf}, \{\mathsf{id}, (8\ 12)(9\ 13)(10\ 14)(11\ 15)\})$ to Γ, where id denotes the identity permutation.
3. The six consecutive coins from c_2 correspond to the three input bits.
4. Apply $(\mathsf{shuf}, \{\mathsf{id}, (1\ 8)(2\ 9)(4\ 10)(6\ 11)\})$ to Γ.
5. Apply $(\mathsf{comp}, \{2, 4, 6\}, \{9, 10, 11\})$ to Γ. If $L = R$ is observed, then output 1 via $(\mathsf{result}, 1)$; otherwise, if this is the second iteration, then output 0 and terminate.
6. Apply $(\mathsf{shuf}, \{\mathsf{id}, (1\ 8)(2\ 9)(4\ 10)(6\ 11)\})$ to Γ again.
7. Apply $(\mathsf{comp}, \{1\}, \{8\})$ to Γ. If $L > R$ is observed, it means that the first four coins correspond to B_{s}. Apply $(\mathsf{shuf}, \{(8\ 12)(9\ 13)(10\ 14)(11\ 15)\})$ to Γ, i.e., exchange the two non-special bag, and return to step 4. Otherwise, apply $(\mathsf{shuf}, \{(1\ 12)(2\ 13)(4\ 14)(6\ 15)\})$ to Γ and return to step 4.

In this way, the protocol is formalized within our model. Based on the above description, any protocol using a pen and bags in such a way can be naturally formalized, such as the existing protocols for any symmetric function and for any Boolean function [25]. The construction of such balance-based protocols without relying on any additional items remains an open problem. Formally, in such a protocol $\mathcal{P}$, the set of permutations Π in any shuffling action (shuf, Π) is required to satisfy $\Pi = S_i$ for the corresponding size $i \leq d$.

6.2 Execution Time

In the proposed balance-based protocol, players place coins one by one, and the execution time increases in proportion to the number of players. A certain number of players can simultaneously place multiple coins on the plate, therefore reducing the execution time. However, there is a limit to such a number, and the execution time would increase as the number of players increases. We evaluate the execution time using a physical balance scale and observe that it takes time for the balance tilt to be determined when the last coin is placed. In other words, to estimate the execution time of balance-based protocols, it is necessary to consider the time required to determine the tilt, and furthermore, it is necessary to verify whether this time varies depending on the weight, which is not obvious.

Card-based cryptography requires a shuffle operation for the AND computation [22], whereas our balance-based protocol shown in Sect. 2.1 only requires a single comparison. Furthermore, our threshold-function protocol requires only one comparison for any number of inputs and does not require any shuffle operation. Since shuffle operations mainly affect the execution time in card-based cryptography [18], our result implies that balance-based protocols may be better

in terms of the execution time than card-based cryptography for certain Boolean functions. However, as mentioned above, precisely estimating the execution time is nontrivial and could be an interesting topic for future research.

7 Conclusion

In this study, we constructed a computational model for balance-based secure computations. This made it possible to rigorously describe balance-based protocols for computing logical AND and majority functions, as well as protocols for computing threshold functions. Our future work includes the modification of the performance of a balance: for example, we may change the balance to tilt only when the weight difference between the two plates exceeds a certain value. This is intended to prevent information leakage that could occur if the speed or degree of tilting reveals the weight difference between the two sides. We also plan to investigate whether the cryptographic protocols developed in this work can be implemented using an actual balance scale. Currently, there remain issues such as potential leakage of weight from the tilting speed or angle of the balance, and we intend to address these problems through further improvements (cf. [29]).

Acknowledgments. We thank the anonymous referees, whose comments have helped us to improve the presentation of the paper. This work was supported in part by JSPS KAKENHI Grant Number JP23H00479, JSPS Bilateral Joint Research Projects JPJSBP120253206, and Institute of Mathematics for Industry, Joint Usage/Research Center in Kyushu University. (FY2025 Short-term Joint Research "Deepening and new frontiers in card-based cryptography through industry-academia collaboration and cooperation in the fields of mathematics and cryptography II" (2025a036)).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Abe, Y., Iwamoto, M., Ohta, K.: Efficient private PEZ protocols for symmetric functions. In: Hofheinz, D., Rosen, A. (eds.) Theory of Cryptography. LNCS, vol. 11891, pp. 372–392. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-36030-6_15
2. Balogh, J., Csirik, J.A., Ishai, Y., Kushilevitz, E.: Private computation using a PEZ dispenser. Theor. Comput. Sci. **306**(1), 69–84 (2003), https://doi.org/10.1016/S0304-3975(03)00210-X
3. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley-Interscience, Hoboken, 2nd edn. (2006), https://doi.org/10.1002/047174882X
4. Crépeau, C., Kilian, J.: Discreet solitary games. In: Stinson, D.R. (ed.) Advances in Cryptology—CRYPTO' 93. LNCS, vol. 773, pp. 319–330. Springer, Berlin, Heidelberg (1994), https://doi.org/10.1007/3-540-48329-2_27
5. Den Boer, B.: More efficient match-making and satisfiability the five card trick. In: Quisquater, J.J., Vandewalle, J. (eds.) Advances in Cryptology—EUROCRYPT '89. LNCS, vol. 434, pp. 208–217. Springer, Berlin, Heidelberg (1990), https://doi.org/10.1007/3-540-46885-4_23

6. Dyson, F.J.: The problem of the pennies. The Mathematical Gazette **30**(291), 231–234 (1946), https://doi.org/10.2307/3611225
7. E.D.Schell: Problem e651-weighed and found wanting. Amer. Math. Monthly **52**, 397 (1945), https://doi.org/10.2307/2304649
8. Gradwohl, R., Naor, M., Pinkas, B., Rothblum, G.N.: Cryptographic and physical zero-knowledge proof systems for solutions of Sudoku puzzles. Theory of Computing Systems **44**(2), 245–268 (2009), https://doi.org/10.1007/s00224-008-9119-9
9. Hanaoka, G.: Towards user-friendly cryptography. In: Phan, R.C.W., Yung, M. (eds.) Paradigms in Cryptology–Mycrypt 2016. Malicious and Exploratory Cryptology. LNCS, vol. 10311, pp. 481–484. Springer, Cham (2017), https://doi.org/10.1007/978-3-319-61273-7_24
10. Kaneko, S., Lafourcade, P., Mallordy, L.B., Miyahara, D., Puys, M., Sakiyama, K.: Balance-based ZKP protocols for pencil-and-paper puzzles. In: Mouha, N., Nikiforakis, N. (eds.) Information Security. LNCS, vol. 15257, pp. 211–231. Springer, Cham (2025), https://doi.org/10.1007/978-3-031-75757-0_11
11. Kaneko, S., Lafourcade, P., Mallordy, L., Miyahara, D., Puys, M., Sakiyama, K.: Secure voting protocol using balance scale. In: Adi, K., Bourdeau, S., Durand, C., Tong, V.V.T., Dulipovici, A., Kermarrec, Y., García-Alfaro, J. (eds.) Foundations and Practice of Security. LNCS, vol. 15532, pp. 365–376. Springer, Cham (2025), https://doi.org/10.1007/978-3-031-87499-4_24
12. Kastner, J., Koch, A., Walzer, S., Miyahara, D., Hayashi, Y., Mizuki, T., Sone, H.: The minimum number of cards in practical card-based protocols. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology—ASIACRYPT 2017. LNCS, vol. 10626, pp. 126–155. Springer, Cham (2017), https://doi.org/10.1007/978-3-319-70700-6_5
13. Koch, A., Schrempp, M., Kirsten, M.: Card-based cryptography meets formal verification. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology–ASIACRYPT 2019. LNCS, vol. 11921, pp. 488–517. Springer, Cham (2019), https://doi.org/10.1007/978-3-030-34578-5_18
14. Koch, A., Walzer, S., Härtel, K.: Card-based cryptographic protocols using a minimal number of cards. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology—ASIACRYPT 2015. LNCS, vol. 9452, pp. 783–807. Springer, Berlin, Heidelberg (2015), https://doi.org/10.1007/978-3-662-48797-6_32
15. Komano, Y., Mizuki, T.: Coin-based secure computations. Int. J. Inf. Secur. **21**, 833–846 (2022), https://doi.org/10.1007/s10207-022-00585-8
16. Marcedone, A., Wen, Z., Shi, E.: Secure dating with four or fewer cards. Cryptology ePrint Archive, Report 2015/1031 (2015), https://eprint.iacr.org/2015/1031
17. Miyahara, D., Komano, Y., Mizuki, T., Sone, H.: Cooking cryptographers: Secure multiparty computation based on balls and bags. In: IEEE Computer Security Foundations Symposium. pp. 1–16. IEEE, NY (2021), https://doi.org/10.1109/CSF51468.2021.00034
18. Miyahara, D., Ueda, I., Hayashi, Y., Mizuki, T., Sone, H.: Evaluating card-based protocols in terms of execution time. Int. J. Inf. Secur. **20**, 729–740 (2021), https://doi.org/10.1007/s10207-020-00525-4
19. Mizuki, T., Komano, Y.: Information leakage due to operative errors in card-based protocols. Inf. Comput. **285**, 104910 (2022), https://doi.org/10.1016/j.ic.2022.104910
20. Mizuki, T., Kumamoto, M., Sone, H.: The five-card trick can be done with four cards. In: Wang, X., Sako, K. (eds.) Advances in Cryptology—ASIACRYPT 2012. LNCS, vol. 7658, pp. 598–606. Springer, Berlin, Heidelberg (2012), https://doi.org/10.1007/978-3-642-34961-4_36

21. Mizuki, T., Shizuya, H.: A formalization of card-based cryptographic protocols via abstract machine. Int. J. Inf. Secur. **13**(1), 15–23 (2014), https://doi.org/10.1007/s10207-013-0219-4
22. Mizuki, T., Sone, H.: Six-card secure AND and four-card secure XOR. In: Deng, X., Hopcroft, J.E., Xue, J. (eds.) Frontiers in Algorithmics. LNCS, vol. 5598, pp. 358–369. Springer, Berlin, Heidelberg (2009), https://doi.org/10.1007/978-3-642-02270-8_36
23. Murata, S., Miyahara, D., Mizuki, T., Sone, H.: Public-PEZ cryptography. In: Susilo, W., Deng, R.H., Guo, F., Li, Y., Intan, R. (eds.) Information Security. LNCS, vol. 12472, pp. 59–74. Springer, Cham (2020), https://doi.org/10.1007/978-3-030-62974-8_4
24. Pass, R., Shelat, A.: A course in cryptography (2010), www.cs.cornell.edu/~rafael/
25. Ruangwises, S.: Balance-based cryptography: Physically computing any Boolean function. In: Unconventional Computation and Natural Computation. LNCS, Springer (2025), to appear
26. Sasaki, T., Miyahara, D., Mizuki, T., Sone, H.: Efficient card-based zero-knowledge proof for Sudoku. Theor. Comput. Sci. **839**, 135–142 (2020), https://doi.org/10.1016/j.tcs.2020.05.036
27. Shinagawa, K., Nuida, K.: Card-based protocols imply PSM protocols. In: Beyersdorff, O., Pilipczuk, M., Pimentel, E., Thàng, N.K. (eds.) Theoretical Aspects of Computer Science. LIPIcs, vol. 327, pp. 72:1–72:18. Schloss Dagstuhl, Dagstuhl (2025), https://doi.org/10.4230/LIPIcs.STACS.2025.72
28. Stiglic, A.: Computations with a deck of cards. Theor. Comput. Sci. **259**(1–2), 671–678 (2001), https://doi.org/10.1016/S0304-3975(00)00409-6
29. Ueda, I., Miyahara, D., Nishimura, A., Hayashi, Y., Mizuki, T., Sone, H.: Secure implementations of a random bisection cut. Int. J. Inf. Secur. **19**(4), 445–452 (2020), https://doi.org/10.1007/s10207-019-00463-w