Computer-Aided Proofs on Impossibility of Card-Based Protocols Using Only Random Cuts

Kakeru Watanabe[1,a)] Shota Ikeda[1] Kazumasa Shinagawa[2,3] Takaaki Mizuki[4]
Kazuki Yoneyama[1]

Abstract: In the research field of card-based cryptography, it is important to find non-trivial lower bounds on the number of cards and shuffles. In particular, many open problems remain for protocols that use only random cuts (RCs) as shuffles, which can be easily demonstrated by human operations. In this paper, we focus on clarifying some lower bounds on the number of cards in protocols using only RCs. First, we develop computer programs to search for states that can be transitioned with RCs and turning over cards. Next, we examine the impossibility of constructing a secure protocol by analyzing the set of states that increase in each step. Based on this approach, our contributions to the specific settings are as follows.

- We revisit Fujita et al.'s proof (APKC 2025) on the impossibility of four-card XOR protocols allowing any number of RCs. We point out and correct errors in their proof.
- At CSS 2019, Abe et al. showed the impossibility of five-card AND protocols as well as a proposal for a six-card AND protocol using only RCs. As their proof on impossibility was partially ambiguous, we show for the first time the impossibility of the five-AND protocols using only RCs under the standard protocol model.
- We discuss for the first time the impossibility of five-card XOR protocols. We show that it is impossible to construct a five-card XOR protocol using ten (or fewer) RCs.

Keywords: card-based cryptography, secure computation, formal methods, proofs on impossibility.

1. Introduction

1.1 Card-Based Cryptography

Card-based cryptography is a general concept that uses physical cards to realize cryptographic techniques, such as secure computations and zero-knowledge proofs. A wide variety of card types is used in the research field of card-based cryptography: the two-colored deck of cards, playing cards, UNO, and so on. This paper focuses on protocols using the two-colored deck of cards, which is typical in card-based cryptography.

Following the conventional practice, we use a two-colored deck of cards, each of which is $\boxed{\clubsuit}$ or $\boxed{\heartsuit}$ on the face and $\boxed{?}$ on the back (in either case). Cards with the same symbols are indistinguishable from each other; in addition, face-down cards are indistinguishable regardless of their symbols. We use the following encoding to represent bits, based on the order of two cards (throughout this paper):

$$\boxed{\clubsuit}\boxed{\heartsuit} = 0, \quad \boxed{\heartsuit}\boxed{\clubsuit} = 1\,. \tag{1}$$

When an actual protocol uses the encoding in (1) to input binary values, the input is hidden by turning over two cards:

$$\boxed{?}\boxed{?}.$$

[1] Ibaraki University
[2] University of Tsukuba
[3] National Institute of Advanced Industrial Science and Technology
[4] Tohoku University
[a)] 24nm771g@vc.ibaraki.ac.jp

Such two face-down cards representing a bit $x \in \{0, 1\}$ are called a *commitment* to x. A protocol whose output is obtained as a commitment is called a *committed-format* protocol, while those not producing a commitment as output are called *non-committed-format* protocols. Committed-format protocols have the advantage of being able to reuse their output as input to another execution of a protocol.

A card-based protocol consists of three main actions: permuting, shuffling, and turning over cards, denoted by perm, shuf, and turn, respectively. Among them, the shuffling action is the most important, ensuring the security of the protocol. Therefore, card-based protocols are often evaluated by the feasibility of implementation and efficiency (i.e., the number of times) of required shuffles. In particular, *random cuts* (RCs) are considered to be the easiest type of shuffles to implement by human operations, and many protocols using only RCs as shuffles have been proposed. With the above background, this paper focuses on committed-format AND and XOR protocols using only RCs.

1.2 Committed-Format AND Protocols Using Only RCs

In 1993, Crépeau-Kilian [1] proposed the first committed-format AND protocol that requires 8 RCs on average using 10 cards of a four-colored deck. As the second protocol that uses only RCs, in 1998, Niemi-Renvall [2] constructed a protocol that requires 7.5 RCs on average using 12 cards of a two-colored deck. Furthermore, in 2001, Stiglic [3] proposed a protocol using 8 cards of a two-colored deck, and in 2021, Abe et al. [4] proposed a protocol using 6 cards of a two-colored deck, each with two RCs. In terms of the impossibility proof, Kastner et al. [5] showed that four-card committed-format AND protocols using only uniform closed shuffles cannot be constructed. Their result encompasses that it is impossible to construct a four-card protocol using only RCs (because any RC is a uniform closed shuffle). Also, at CSS 2019, Abe et al. [6] showed the impossibility of five-card AND protocols. However, their proof on impossibility was partially ambiguous.

1.3 Committed-Format XOR Protocols Using Only RCs

In 1993, Crépeau-Kilian [1] proposed the first committed-format XOR protocol that requires 10 RCs (on average) using 14 cards of a four-colored deck. As the second committed-format XOR protocol that uses only RCs, in 2006, Mizuki et al. [7] proposed a protocol using only 7 RCs (on average) with 10 cards of a two-colored deck. In addition, in 2020, Toyoda et al. [8] proposed a committed-format XOR protocol that requires 2 RCs using 6 cards of a two-colored deck. Furthermore, in 2025, Fujita et al. [9] showed that it is impossible to construct a committed-format XOR protocol using four cards of a two-colored deck with only RCs. Even so, it is still a remaining problem whether it is (im)possible to construct a committed-format XOR protocol using five cards of a two-colored deck with only RCs.

1.4 Our Contributions

In this paper, we tackle the above open problems by developing C# (.NET environment) applications to search for states transitioned from an initial state with only RCs and turn. More specifically, we analyze how a state increases from a single RC and any number of turn actions, and consider the impossibility of a protocol by examining the feature of state increase. Based on these methods and ideas, we obtain the following results.

1.4.1 Revisiting Existing Proof on Impossibility of Four-Card XOR Protocols

At APKC 2025, Fujita et al. [9] showed the impossibility of four-card committed-format XOR protocols based on two methods: formal methods and proof using pen and paper. We focus on their proof using pen and paper that discusses the impossibility where any number of operations are allowed. Their proof claimed that ultimately, there would be no further operations that could be applied (dead ends). However, we show that in fact there are still operations that can be applied even after reaching the dead ends they claimed, and that it is impossible to reach a dead end after that. Our modified proof shows that even if any number of operations are allowed, four-card committed-format XOR protocols still cannot be constructed.

1.4.2 Impossibility of Five-Card AND Protocols

At CSS 2019, Abe et al. [6] showed the impossibility of five-card AND protocols as well as a proposal for a six-card AND protocol using only RCs. However, we believe that there are several ambiguities in their proofs. Hence, we give more rigorous proofs in this paper. In the five-card AND protocols, if we repeat the operations that can be applied to the initial state, we show that all states newly created after the 5th RC are non-turnable states, and any turnable states do not appear even in the 6th RC and thereafter. Our asymptotic evaluation shows that, assuming a five-card AND protocol, it is impossible to reach a final state even if any number of operations are allowed, and therefore, such a protocol cannot be constructed. Our result leads us to conclude that the existing six-card AND protocol proposed by Abe et al. [4] is optimal in terms of the number of cards.

1.4.3 Impossibility of Five-Card XOR Protocols

We also prove the impossibility of a five-card XOR protocol. The number of states appearing in an XOR protocol is more rapidly increased compared to the case of AND protocols, and hence, we cannot analyze it in the same way as the AND protocol. Our analysis shows that assuming a five-card XOR protocol, the final state does not appear using 10 (or fewer) times of RCs, and therefore, such an XOR protocol cannot be constructed.

2. Preliminaries

This paper deals with protocols based on the Mizuki-Shizuya model [10], which is a standard model for card-based cryptography.

2.1 Operations

In the Mizuki-Shizuya model, the operations that can be applied to card sequences are turn, perm, and shuf.

- turn. The operation of turning a card is called turn: for a sequence of N cards, the set of card positions $T \in \{1, 2, ..., N\}$ specifies the cards to be turned over, resulting in turning over every i th $\in T$ card.
- perm. The operation of permuting cards is called perm, and it takes a permutation $\pi \in S_N$, which means that the sequence of N cards are permuted according to π.
- shuf. The operation of shuffling a sequence of cards is called shuffle, which is a generalization of perm. For a sequence of N cards, two sets are defined: the permutation set $\Pi \subseteq S_N$ and the probability distribution $\mathcal{F}$ on Π. These sets are called shuffle pairs. A permutation $\pi \in \Pi$ is probabilistically chosen according to $\mathcal{F}$, and π is applied to the sequence.

2.2 Random Cuts (RCs)

A shuffle with a permutation set Π and a probability distribution $\mathcal{F}$ is called a random cut (RC) if Π is a cyclic group generated by a single loop σ and $\mathcal{F}$ is the uniform distribution on Π. We denote a random cut defined by a single loop σ by RC_{σ}. For example, a random cut $\mathsf{RC}_{(1,2,3)}$ is a shuffle operation that chooses a permutation π from $\Pi = \{\mathsf{id}, (1, 2, 3), (1, 3, 2)\}$ uniformly at random, and then rearranges the sequence of cards according to π.

There are several ways to implement random cuts. One method that does not require additional tools is using the Hindu cut [11]. Another method involves using a rotating plate that can hold cards. Since these implementation methods are easy to implement, random cuts are considered one of the most basic shuffles.

2.3 KWH-tree

A well-known visual representation method for card-based protocols is the KWH-tree proposed by Koch et al. [12]. In the KWH-tree, the state is represented by a box, with arrows drawn out for each operation applied to it. Each state holds the card sequence and its probability distribution in that state, and X_{ab} appearing in each state represents a random variable whose input value is $(a, b) \in \{0, 1\}^2$. For each state and its transitions, there is no bifurcation due to the perm and shuf operations, but the turn operation causes a bifurcation depending on the suit.

A state is defined as a function $\mu : \mathcal{S} \to P$ from the whole set of card sequences to the set of homogeneous polynomials over X_{ab} $(a, b \in \{0, 1\})$ of degree 1 with non-negative coefficients:

$$P := \{p_{00}X_{00} + p_{01}X_{01} + p_{10}X_{10} + p_{11}X_{11} | p_{ij} \geq 0\}.$$

2.4 Turnability of a State

A state μ is *turnable* for $i \in \{1, 2, ..., N\}$ if for any $c \in \{\heartsuit, \clubsuit\}$, there exists a real number $0 \leq p \leq 1$ such that the equation (2) holds:

$$\sum_{s \in \mathcal{S}, s[i]=c} \mu(s) = p(X_{00} + X_{01} + X_{10} + X_{11}), \qquad (2)$$

where $s[i]$ is the symbol of the i-th card of the sequence s. When a state is turnable, it seems intuitive to say that the informational content of any input is not leaked by the resulting suit of cards turned over.

2.5 ⊥-sequence

If the probabilities of inputs with different outputs within a state are mixed in a single sequence of cards, the state cannot become a final state, no matter what operations are subsequently repeated. Such a sequence of cards is called a $\perp$-sequence. Specifically, a card sequence $s \in \mathcal{S}$ that has the probability $\mu(s) = \sum_{a,b} p_{ab}X_{ab}$ satisfying the following equation have a $\perp$-sequence:

$$(p_{00} + p_{01} + p_{10} > 0) \wedge (p_{11} > 0) \text{ for AND Protocol,}$$
$$(p_{00} + p_{11} > 0) \wedge (p_{01} + p_{10} > 0) \text{ for XOR Protocol.}$$

2.6 i/j–state

As a designation for a state, we use the designation i/j-state in this paper. An i/j-state is a state with i sequences that output 0 and j sequences that output 1 within that state. Also, the trivial property here is that if either i or j is equal to 1, the state does not satisfy turnability. This is because i and j must be greater than 1 in all states to which they are turned.

Furthermore, by extending i/j-states to $i/j/k/l$-states, states where any of i, j, k or l is equal to 1 do not satisfy turnability for the same reason. Here i, j, k, l are the number of sequences corresponding to each input, such as X_{00}, X_{01}, X_{10}, and X_{11}.

2.7 Final State

The conditions for becoming a final state in a committed-format protocol are as follows. First, each sequence in state is split into a set of sequences outputting 0 ($= S_{s\mapsto 0}$) and a set of sequences outputting 1 ($= S_{s\mapsto 1}$). Then create index sets $S_{0\clubsuit}$, $S_{0\heartsuit}$, $S_{1\clubsuit}$ and $S_{1\heartsuit}$ where $\clubsuit/\heartsuit$ appears in common from $S_{s\mapsto 0}$ and $S_{s\mapsto 1}$. Finally, the condition for a final state is the existence of an index that appears in common among $S_{0\clubsuit}$ and $S_{1\heartsuit}$, and $S_{0\heartsuit}$ and $S_{1\clubsuit}$.

The intuitive explanation is that the condition for the final state is that the commitment can be made symmetrically in each of the card sequences that output 0 or 1.

3. Computer-Aided State Search

See (`github.com/xk-watanabe/Impossibility_Card_RCs`) for our codes.

3.1 Overall Strategy

Our strategy for searching final states starting from the initial state is described as follows.

(1) Set $S, S_0 \leftarrow \{\mu_0\}$ for the initial state μ_0.

(2) Do the following procedure.

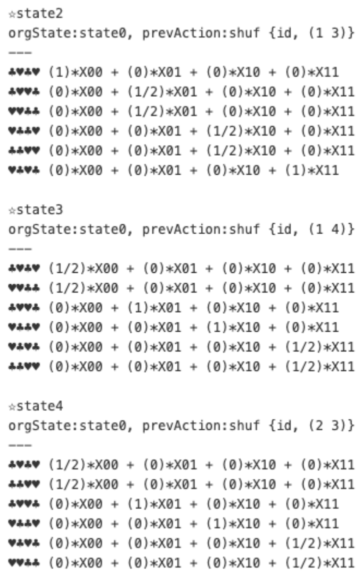
- (a) Set $S_1, S_2 \leftarrow \emptyset$.
- (b) For each $\mu \in S_0$, apply an applicable RC to μ and obtain a new state μ'. Set $S_1 \leftarrow S_1 \cup \{\mu'\}$. This step is performed for all applicable RCs.
- (c) For each $\mu \in S_1$, apply an applicable turn operation to μ and obtain new states $\mu'_\clubsuit, \mu'_\heartsuit$. Update $S_1 \leftarrow S_1 \cup \{\mu'_\clubsuit, \mu'_\heartsuit\}$ and $S_2 \leftarrow S_2 \cup \{\mu'_\clubsuit, \mu'_\heartsuit\}$. This step is performed until no new states are added to S_2. Note that this step deals with a case where two or more turn operations are performed sequentially for a state $\mu \in S_1$.
- (d) If $S_1 = S_2 = \emptyset$, terminate the search.
- (e) For each state $\mu \in S_2$, it is flagged if μ is final.
- (f) Update $S_0 \leftarrow S_1 \cup S_2$ and $S \leftarrow S \cup S_1 \cup S_2$. Write the all states in S to an external file and go to Step (2a) for the next loop.

Here, it should be explained that why we apply only RCs and turn operations in Steps (2b) and (2c), and ignore perm operations. This is because applying (perm, π) and then RC_σ is the same as applying $\mathsf{RC}_{\pi^{-1}\sigma\pi}$ and then (perm, π), and similarly, applying (perm, π) and then (turn, $\{i\}$) is the same as applying (turn, $\{\pi^{-1}(i)\}$) and then (perm, π). Thus, all perm operations can be moved to the last step of the protocol, and the perm operation at the last step can be removed because applying perm operations does not change whether a state is final.

It should be also explained that in Step (e), why we check whether a state is final for S_2 only. This is because applying a shuf operation to a non-final state does not result in a final state. Since the initial state is clearly not final, so we only need to check whether the states after turn operations are final.

We implement the above strategy using C#. In this state search, each state is managed as a class on C#, with the following information: the original state from which it transitioned and its operation, its state number, a flag to indicate whether it is the final state (default value: false), and card sequences (including the coefficient for each input). If a certain operation is performed on a state as described above and the result is equivalent to an existing state, information on the original state and the operation performed on it is added to the existing state.

For the last step in each loop, our program outputs all newly found states to a file as a log. Figure 1 is an example of the log of the four-card XOR protocol. As you can see, for each state, we know the original state and the operation on it, as well as the card sequences and probability distributions.



```
☆state2
orgState:state0, prevAction:shuf {id, (1 3)}
---
♣♥♣♥ (1)*X00 + (0)*X01 + (0)*X10 + (0)*X11
♣♥♥♣ (0)*X00 + (1/2)*X01 + (0)*X10 + (0)*X11
♥♥♣♣ (0)*X00 + (1/2)*X01 + (0)*X10 + (0)*X11
♥♣♣♥ (0)*X00 + (0)*X01 + (1/2)*X10 + (0)*X11
♣♣♥♥ (0)*X00 + (0)*X01 + (1/2)*X10 + (0)*X11
♥♣♥♣ (0)*X00 + (0)*X01 + (0)*X10 + (1)*X11

☆state3
orgState:state0, prevAction:shuf {id, (1 4)}
---
♣♥♣♥ (1/2)*X00 + (0)*X01 + (0)*X10 + (0)*X11
♥♥♣♣ (1/2)*X00 + (0)*X01 + (0)*X10 + (0)*X11
♣♥♥♣ (0)*X00 + (1)*X01 + (0)*X10 + (0)*X11
♥♣♣♥ (0)*X00 + (0)*X01 + (1)*X10 + (0)*X11
♥♣♥♣ (0)*X00 + (0)*X01 + (0)*X10 + (1/2)*X11
♣♣♥♥ (0)*X00 + (0)*X01 + (0)*X10 + (1/2)*X11

☆state4
orgState:state0, prevAction:shuf {id, (2 3)}
---
♣♥♣♥ (1/2)*X00 + (0)*X01 + (0)*X10 + (0)*X11
♣♣♥♥ (1/2)*X00 + (0)*X01 + (0)*X10 + (0)*X11
♣♥♥♣ (0)*X00 + (1)*X01 + (0)*X10 + (0)*X11
♥♣♣♥ (0)*X00 + (0)*X01 + (1)*X10 + (0)*X11
♥♣♥♣ (0)*X00 + (0)*X01 + (0)*X10 + (1/2)*X11
♥♥♣♣ (0)*X00 + (0)*X01 + (0)*X10 + (1/2)*X11
```

Fig. 1 An Example of Output Log of Our State Search.

3.2 Initial State

Here, we define an N-card protocol means that no cards other than the N cards in the initial state are used. In other words, no cards are added or removed during the protocol.

3.2.1 For Four-Card Protocols

As for the four-card protocol, since two cards are used per input with two inputs, it is trivially settled on one state. The state is marked as Table 1.

3.2.2 For Five-Card Protocols

As for the five-card protocol, one additional card must be given in addition to each input. There are many possible patterns for the choice of suits and the position of the additional card, but it is sufficient to consider one particular way of doing this. First, in terms of the symmetry of suit selection, for each case of adding $\boxed{♣}$ and $\boxed{♡}$, the same state of affairs can be obtained by reversing the suits of all cards. Also, as noted in Section 3.1, protocol operations that use perm can be converted to protocols that do not use perm. In other words, the results of the impossibility discussions for a protocol with an additional card in one location can be translated through the concept of perm to the results for an additional card in another location. Thus, it is sufficient to consider the impossibility of the five-card protocols if we consider the case of adding a card of a certain suit for one location. In this paper, we consider the case of adding a ♡ card in the center for the Table 1 case in a four-card setup. The specific initial state is shown as Table 2.

Table 1 state μ_0^4

♣♡♣♡	X_{00}
♣♡♡♣	X_{01}
♡♣♣♡	X_{10}
♡♣♡♣	X_{11}

Table 2 state μ_0^5

♣♡♡♣♡	X_{00}
♣♡♡♡♣	X_{01}
♡♣♡♣♡	X_{10}
♡♣♡♡♣	X_{11}

3.3 Visualization of KWH-tree with QuickGraph

We visualize KWH-trees in the four-card XOR protocol and the five-card AND protocol using QuickGraph (a library for handling graph structures for .NET) in order to study the increasing trend of states. This allows us to expect a visually-based understanding of the relationships between states, which is difficult to read from the logs alone. Specific output examples are shown in Section 4.

3.4 State Group

Here, we newly define a *state group* as a relationship between states. In this paper, a state group is a set of states in which all the correspondences between inputs and card sequences within each state are matched, and the result of applying applicable operations to them to create new states (non-existing states) is that they also have the same correspondences between inputs and card sequences as the original state. In particular, as a result of outputting the KWH-tree as a result of the state search, this paper focuses on the fact that states with the above relationship only continue to change their probability coefficients, and the relationship as a state group does not change even if the operation is applied continuously, and discusses the impossibility based on this.

4. Revisiting Fujita et al.'s Proof on Impossibility of Four-Card XOR Protocols

4.1 Result of Our State Search

The result of the state search for four-card XOR protocols is presented as Table 3. By applying a RC to the initial state, we can see that 6 states are newly generated. By applying a turn operation to the 6 states, we obtain 16 states. After applying the second RC to $(6+16)$ states, we obtain 46 states and none of them are turnable. After applying the third RC to 46 states, we obtain 48 states and none of them are turnable. After the fourth RC, the 48 states will produce new 48 states, which are not turnable, and this process continues infinitely.

Figure 2 shows a graphical representation of generation of states. The left figure shows that, in the third RC, three orange states are newly generated from two blue states. Similarly, the right figure shows that, in the fourth RC, three orange states are newly generated from three blue states. In both the third and fourth RCs, there are 16 groups of such generation, and thus 48 states are newly generated.

Table 3 Result of State Search of Four-Card XOR Protocols

# of Shuffle	Final State?	Total of States	+ Shuffle	+ Turn
1	No	23	6	16
2	No	69	46	0
3	No	117	48	0
4	No	165	48	0
5	No	213	48	0
6	No	261	48	0
7	No	309	48	0
8	No	357	48	0
9	No	405	48	0
10	No	453	48	0

"Final State?" indicates whether there is a final state among the states searched so far, "Total of States" indicates the total number of states searched so far, and "+ Shuffle" and "+ Turn" indicate how many new states were found by shuffle operation and turn operation, respectively.

4.2 Errors in Fujita et al.'s Proof

Fujita et al.'s proof shows that the number of states obtained

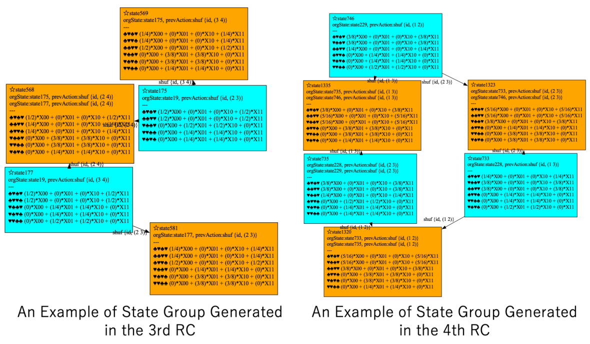

An Example of State Group Generated in the 3rd RC
An Example of State Group Generated in the 4th RC

Fig. 2 A Part of State Groups Generated in State Search for Four-Card XOR Protocols.

from applicable operations is finite, and none of them are final. However, their proof is incorrect because we obtain an infinite number of states from applicable operations. Indeed, for the blue states on the right side of Figure 2, there are several RCs that are applicable to them. Here, after the 3rd RC, the 48 blue states will generate 48 new states, and it continues infinitely (see Table 3).

Note that, except for the 48 states, Fujita et al.'s proof is correct. Based on the argument in their proof, the 48 states are grouped into 16 groups of 3 states, and all groups are symmetric. Due to the symmetry, if 3 states of a group do not produce a final state, it implies that all 48 states do not produce a final state. Thus, to obtain complete proof, it suffices to show that none of the states obtained from 3 states of a group are final.

4.3 Our Complete Proof

In this section, we prove the impossibility of the four-card committed-format XOR protocol.

Theorem 1. *There exists no four-card committed-format XOR protocol using only random cuts.*

Proof. Based on the Fujita et al.'s proof with the discussion in Section 4.2, it is sufficient to show that 3 states of a group after the fourth RC do not produce a final state. For example, μ_{13}, μ_{14}, and μ_{15} in Table 8, 9, and 10, respectively, form such a group.

Let μ_0 be one of three states μ_{13}, μ_{14}, and μ_{15}. Set each coefficient of μ_0 as $a_0, b_0, \dots, f_0$ as in Table 11. Now we can observe that for any state of the form in Table 11, there are only three applicable operations: $\mathsf{RC}_{(1,2)}$ referred to as operation a, $\mathsf{RC}_{(2,4)}$ referred to as operation b, and $\mathsf{RC}_{(1,2,4)}$ referred to as operation c. In addition, applying these RCs to the state in Table 11 results in the states in Tables 12, 13, and 14, and they are still of the form in Table 11.

Starting from μ_0, let μ_n be a state obtained from μ_{n-1} by applying one of the three applicable operations. Set each coefficient of μ_n as $a_n, b_n, \dots, f_n$ as in Table 11. Then we have a sequence of states $\mu_0, \mu_1, \mu_2, \dots, \mu_n$. We need to show that each state μ_n for $n \geq 0$ is not turnable. Note that the 3rd card cannot be trivially turnable. Here, the state μ_n is turnable if and only if one of the following equation pairs hold:

$$\begin{cases} a_n = d_n + f_n & \text{(when the 1st card is } \heartsuit) \quad (3) \\ b_n + c_n = e_n & \text{(when the 1st card is } \clubsuit) \quad (4) \end{cases}$$

$$\begin{cases} c_n = e_n + f_n & \text{(when the 2nd card is } \heartsuit) \quad (5) \\ a_n + b_n = d_n & \text{(when the 2nd card is } \clubsuit) \quad (6) \end{cases}$$

$$\begin{cases} a_n + c_n = f_n & \text{(when the 4th card is } \heartsuit) \quad (7) \\ b_n = d_n + e_n & \text{(when the 4th card is } \clubsuit) \quad (8) \end{cases}$$

Thus, we need to show that all of the above equations do not hold.

Lemma 1. *For any $n \geq 0$, Equations (3) to (8) do not hold.*

Proof. We will show that Equation (3) does not hold for any $n \geq 0$. Other cases can be proven by a similar way.

First, we can easily observe that Equation (3) does not hold for $n = 0$ from Tables 8, 9, and 10. We suppose by contradiction that $n > 0$ is the minimum number satisfying $a_n \neq d_n + f_n$.

Since the summation of the coefficients of the probability of each input sums to 1, the following equations hold:

$$\begin{cases} a_n + b_n + c_n &= 1, \\ d_n + e_n + f_n &= 1. \end{cases} \quad (9)$$

Also, since each coefficient is non-negative, the following relationship also holds:

$$a_n, b_n, c_n, d_n, e_n, f_n > 0. \quad (10)$$

In the base stage, the proposition holds in μ_{13}, μ_{14} and μ_{15}. We prove it by contradiction. Suppose that $n > 0$ is the minimum number satisfying the equation $a_n = d_n + f_n$.

If the n_{th} operation is a: In this case, we have $a_n = \frac{1}{2}(a_{n-1}+c_{n-1})$, $d_n = \frac{1}{2}(d_{n-1}+e_{n-1})$, and $f_n = f_{n-1}$, from Table 12. By substituting them into $a_n = d_n + f_n$, we have:

$$\frac{1}{2}(a_{n-1} + c_{n-1}) = \frac{1}{2}(d_{n-1} + e_{n-1}) + f_{n-1}.$$

From Equation (9), we have:

$$\frac{1}{2}(1 - b_{n-1}) = \frac{1}{2}(1 + f_{n-1}).$$

This implies that $-b_{n-1} = f_{n-1}$, which contradicts (10).

If the n_{th} operation is b: In this case, we have $a_n = a_{n-1}$, $d_n = \frac{1}{2}(d_{n-1} + f_{n-1})$, and $f_n = \frac{1}{2}(d_{n-1} + f_{n-1})$, from Table 13. By substituting them into $a_n = d_n + f_n$, we have:

$$a_{n-1} = \frac{1}{2}(d_{n-1} + f_{n-1}) + \frac{1}{2}(d_{n-1} + f_{n-1}).$$

This implies that $a_{n-1} = d_{n-1} + f_{n-1}$, which contradicts the minimality of n satisfying the equation $a_n = d_n + f_n$.

If the n_{th} operation is c: In this case, we have $a_n = \frac{1}{3}(a_{n-1}+b_{n-1}+c_{n-1})$, $d_n = \frac{1}{3}(d_{n-1} + e_{n-1} + f_{n-1})$, and $f_n = \frac{1}{3}(d_{n-1} + e_{n-1} + f_{n-1})$, from Table 14. By substituting them into $a_n = d_n + f_n$, we have:

$$\frac{1}{3}(a_{n-1}+b_{n-1}+c_{n-1}) = \frac{1}{3}(d_{n-1}+e_{n-1}+f_{n-1})+\frac{1}{3}(d_{n-1}+e_{n-1}+f_{n-1})$$

From Equation (9), we have:

$$\frac{1}{3} = \frac{2}{3}.$$

This is not mathematically valid.

From the above, the assumption is incorrect because a contradiction was derived for any of the operations. Therefore $a_n \neq d_n + f_n$. □

In a similar way to the proof of Lemma 1, it can also be shown

Table 4 Summary of Contradictions in Each of the Equations

Assumption	Operation a	Operation b	Operation c
$a_n = d_n + f_n$	$b_{n-1} = -f_{n-1}$	$a_{n-1} = d_{n-1} + f_{n-1}$	$1/3 = 2/3$
$b_n + c_n = e_n$	$b_{n-1} = -f_{n-1}$	$b_{n-1} + c_{n-1} = e_{n-1}$	$2/3 = 1/3$
$a_n + b_n = d_n$	$b_{n-1} = -f_{n-1}$	$a_{n-1} = -e_{n-1}$	$2/3 = 1/3$
$c_n = e_n + f_n$	$b_{n-1} = -f_{n-1}$	$a_{n-1} = -e_{n-1}$	$1/3 = 2/3$
$a_n + c_n = f_n$	$a_{n-1} + c_{n-1} = f_{n-1}$	$a_{n-1} = -e_{n-1}$	$2/3 = 1/3$
$b_n = d_n + e_n$	$b_{n-1} = d_{n-1} + e_{n-1}$	$a_{n-1} = -e_{n-1}$	$1/3 = 2/3$

that Equations (4) to (8) do not hold for any $n \geq 0$. Table 4 summarizes the contradictions obtained from each assumption. In Table 4, we assume that n is the minimum number satisfying the equation in the column "Assumption". From Table 4, we find that μ_{13}, μ_{14}, and μ_{15} do not produce a turnable state, regardless of the number of operations applied. Therefore, the impossibility of the four-card committed-format XOR protocol is proven. □

5. Proof on Impossibility of Five-Card AND Protocols

5.1 Result of State Search and Our Analysis

The result of the state search for five-card AND protocols is presented as Table 5. As Table 5 shows, after the 5th RC, 114 new states are generated by RC, and no new states are generated by turn operations. These 114 states generated in the 5th RC can be classified into the following classes.

- **6/3-states:** 10 state groups of 3 states (Total: 30 states)
- **3/6-states:** 10 state groups of 3 states (Total: 30 states)
- **7/3-states:** 6 state groups of 3 states and 3 state groups of 6 states (Total: 36 states)
- **9/1-states:** 3 state groups of 6 states (Total: 18 states)

For the 6/3-states and 3/6-states, we find that the correspondence between the input and the card sequence all match by doing some perm between their respective state groups. Therefore, it is sufficient to be able to abstract probability coefficients and derive impossibilities for one of state group, respectively.

Table 5 Result of State Search of Five-Card AND Protocols

# of Shuffle	Final State?	Total of States	+ Shuffle	+ Turn
1	No	28	17	10
2	No	175	102	45
3	No	442	237	30
4	No	582	135	5
5	No	696	114	0
6	No	810	114	0
7	No	924	114	0
8	No	1038	114	0
9	No	1152	114	0
10	No	1266	114	0

5.2 Our Proof on Impossibility

In this section, we prove the impossibility of the five-card committed-format AND protocol.

Theorem 2. *There exists no five-card committed-format AND protocol using only random cuts.*

Proof. Based on our analysis in Section 5.1, we will show the impossibility for each state class, respectively.

Lemma 2. *A turnable state cannot be generated from 6/3-states.*

Proof. An example of an original 6/3-state group in the 5th RC is shown as Table 15, 16 and 17. Let μ_0 be one of three states μ_x, μ_y, and μ_z. Set each coefficient of μ_0 as a_0, b_0, ..., i_0 as Table 18. Now we can observe that for any state of the form in Table 18, there are only four applicable operations: $\mathsf{RC}_{(1,4)}$ referred to as operation a, $\mathsf{RC}_{(1,5)}$ referred to as operation b, $\mathsf{RC}_{(4,5)}$ referred to as operation c, and $\mathsf{RC}_{(1,4,5)}$ referred to as operation d. Note that operation d is an operation that does not generate a new state. In addition, applying these RCs to the state in Table 18 results in the states in Tables 19, 20, and 21, and they are still of the form in Table 18. Starting from μ_0, let μ_n be a state obtained from μ_{n-1} by applying one of the three applicable operations. Set each coefficient of μ_n as $a_n, b_n, \ldots, i_n$ as in Table 18. Then we have a sequence of states $\mu_0, \mu_1, \mu_2, \ldots, \mu_n$. We need to show that each state μ_n for $n \geq 0$ is not turnable. Note that the 2nd and 3rd cards cannot be trivially turnable. Here, the state μ_n is turnable if and only if one of the following equation pairs hold:

$$\begin{cases} a_n + b_n + d_n + e_n = g_n & \text{(when the 1st card is } \heartsuit) \quad (11) \\ c_n + f_n = h_n + i_n & \text{(when the 1st card is } \clubsuit) \quad (12) \end{cases}$$

$$\begin{cases} a_n + c_n + d_n + f_n = h_n & \text{(when the 4th card is } \heartsuit) \quad (13) \\ b_n + e_n = g_n + i_n & \text{(when the 4th card is } \clubsuit) \quad (14) \end{cases}$$

$$\begin{cases} b_n + c_n + e_n + f_n = i_n & \text{(when the 5th card is } \heartsuit) \quad (15) \\ a_n + d_n = g_n + h_n & \text{(when the 5th card is } \clubsuit) \quad (16) \end{cases}$$

Thus, we need to show that all of the equations (11) to (16) do not hold. Since the summation of the coefficients of the probability of each input sums to 1, the following equations hold:

$$\begin{cases} a_n + b_n + c_n + d_n + e_n + f_n &= 1, \\ g_n + h_n + i_n &= 1. \end{cases} \qquad (17)$$

Also, since each coefficient is non-negative, the following inequality also holds:

$$a_n, b_n, c_n, d_n, e_n, f_n, g_n, h_n, i_n > 0. \qquad (18)$$

In the base stage, the proposition holds in μ_x, μ_y and μ_z. We prove it by contradiction. Suppose that $n > 0$ is the minimum number satisfying the equations (11) to (16). This can be shown by the similar procedure as for Lemma 1 that equations (11) to (16) cannot hold using relations (17) and (18). Table 7 summarizes the contradictions obtained from each assumption. Therefore, it is proven that a turnable state cannot be generated from the 6/3-states. □

Lemma 3. *A turnable state cannot be generated from 3/6-states.*

Proof. It can be similarly shown as in Lemma 2 that no turnable state can be obtained from 3/6-states. This is omitted in this paper. □

Lemma 4. *A turnable state cannot be generated from 7/3-states.*

Proof. Every 7/3-state appeared after the 5th RC is a state where either i, j or k is 1 when this is extended to $i/j/k/l$-states. According to our state search, the results of applying the applicable RC to the 7/3-state all match the original state in terms of the correspondence between the input and the card sequence. Hence, the 7/3-state cannot satisfy turnability because it is obvious that information about the input is leaked when a turn is performed. □

Lemma 5. *A turnable state cannot be generated from 9/1-states.*

Proof. If 9/1-state generates only 9/1-state, then this one also does not satisfy turnability as same reason of Lemma 4. □

From Lemmas 2 to 5, applicable operations after the 6th RC can yield no turnable state and hence no final state. Therefore, the impossibility of the five-card committed-format AND protocol is proven. □

6. Impossibility of Five-Card XOR Protocols

Table 6 Result of State Search of Five-Card XOR Protocols

# of Shuffle	Final State?	Total of States	+ Shuffle	+ Turn	Exec. Time(ms)
1	No	53	16	36	140
2	No	549	366	130	260
3	No	2205	1412	244	1195
4	No	6227	3718	304	8186
5	No	14631	8044	360	57585
6	No	31667	16676	360	453534
7	No	65279	33252	360	2957398
8	No	131635	65996	360	15066635
9	No	263903	131908	360	68641175
10	No	530287	266024	360	299987681

The result of the state search for five-card XOR protocols is shown in Table 6. This state search was conducted on an Intel Core i9-11900K with 128 GB of RAM. As Table 6 shows, as the number of RC increases, the number of newly generated states also increases, so it seems difficult to show the impossibility using a similar argument as for the four-card XOR protocol and five-card AND protocol.

For the above reasons, this paper declines to discuss the impossibility of allowing any number of operations of the five-card XOR protocol. However, our result leads us to the impossibility of five-card XOR protocols using 10 or fewer times of RCs.

7. Conclusion

This paper showed the impossibility for each of the four-card XOR, five-card AND, and five-card XOR protocols using only RCs. In particular, it was shown that for the four-card XOR and five-card AND protocols, it is impossible to construct a secure protocol after applying an arbitrary number of operations, and for the five-card XOR protocol, it is impossible to construct a protocol using 10 or fewer times of RCs.

It is an important open problem to prove the impossibility of five-card XOR protocols when any number of operations are allowed. Another open problem is to show the optimality of existing protocols. Abe et al.'s [4] six-card AND protocol and Toyoda et al.'s [8] six-card XOR protocol use two RCs, but it is not known whether these protocols can be constructed using only one RC. In addition, these protocols use $\boxed{\clubsuit}\boxed{\heartsuit}$ as additional cards, and it is a future work to clarify whether the protocol can be constructed by the same or fewer number of RCs even if the additional cards are cards of the same color, such as $\boxed{\clubsuit}\boxed{\clubsuit}$ or $\boxed{\heartsuit}\boxed{\heartsuit}$.

Acknowledgments This work was supported by JST CREST Grant Number JPMJCR22M1, JSPS KAKENHI Grant Numbers JP21K17702, JP23H00479, and JP24K02938, and Institute of Mathematics for Industry, Joint Usage/Research Center in Kyushu University. (FY2023 Short-term Joint Research "Organizing open problems in card-based cryptography through industry-academia collaboration" (2023a020), and 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))

References

[1] Crépeau, C. and Kilian, J.: Discreet Solitary Games, *Advances in Cryptology—CRYPTO' 93* (Stinson, D. R., ed.), LNCS, Vol. 773, Berlin, Heidelberg, Springer, pp. 319–330 (online), available from ⟨https://doi.org/10.1007/3-540-48329-2_27⟩ (1994).

[2] Niemi, V. and Renvall, A.: Secure multiparty computations without computers, *Theor. Comput. Sci.*, Vol. 191, No. 1–2, pp. 173–183 (online), available from ⟨https://doi.org/10.1016/S0304-3975(97)00107-2⟩ (1998).

[3] Stiglic, A.: Computations with a Deck of Cards, *Theor. Comput. Sci.*, Vol. 259, No. 1–2, pp. 671–678 (online), available from ⟨https://doi.org/10.1016/S0304-3975(00)00409-6⟩ (2001).

[4] Abe, Y., Mizuki, T. and Sone, H.: Committed-format AND protocol using only random cuts, *Nat. Comput.*, Vol. 20, No. 4, pp. 639–645 (online), available from ⟨https://doi.org/10.1007/s11047-021-09862-2⟩ (2021).

[5] Kastner, J., Koch, A., Walzer, S., Miyahara, D., Hayashi, Y., Mizuki, T. and Sone, H.: The Minimum Number of Cards in Practical Card-Based Protocols, *Advances in Cryptology—ASIACRYPT 2017* (Takagi, T. and Peyrin, T., eds.), LNCS, Vol. 10626, Cham, Springer, pp. 126–155 (online), available from ⟨https://doi.org/10.1007/978-3-319-70700-6_5⟩ (2017).

[6] Abe, Y., Mizuki, T. and Sone, H.: Improved Committed AND Protocol Using Only Random Cut and Impossibility of Reducing the Number of Cards (in Japanese), *Computer Security Symposium 2019*.

[7] Mizuki, T., Uchiike, F. and Sone, H.: Securely computing XOR with 10 cards, *The Australasian Journal of Combinatorics*, Vol. 36, pp. 279–293 (online), available from ⟨https://ajc.maths.uq.edu.au/?page=get_volumes&volume=36⟩ (2006).

[8] Toyoda, K., Miyahara, D., Mizuki, T. and Sone, H.: Six-Card Finite-Runtime XOR Protocol with Only Random Cut, *ACM Workshop on ASIA Public-Key Cryptography*, New York, ACM, pp. 2–8 (online), available from ⟨https://doi.org/10.1145/3384940.3388961⟩ (2020).

[9] Fujita, K., Ikeda, S., Shinagawa, K. and Yoneyama, K.: Formal Verification and Proof of Impossibility for Four-Card XOR Protocols Using Only Random Cuts, *ACM Workshop on ASIA Public-Key Cryptography*, ACM, pp. 9–16 (online), available from ⟨https://doi.org/10.1145/3709015.3728665⟩ (2025).

[10] Mizuki, T. and Shizuya, H.: A formalization of card-based cryptographic protocols via abstract machine, *Int. J. Inf. Secur.*, Vol. 13, No. 1, pp. 15–23 (online), available from ⟨https://doi.org/10.1007/s10207-013-0219-4⟩ (2014).

[11] Ueda, I., Nishimura, A., Hayashi, Y., Mizuki, T. and Sone, H.: How to Implement a Random Bisection Cut, *Theory and Practice of Natural Computing* (Martín-Vide, C., Mizuki, T. and Vega-Rodríguez, M. A., eds.), LNCS, Vol. 10071, Cham, Springer, pp. 58–69 (online), available from ⟨https://doi.org/10.1007/978-3-319-49001-4_5⟩ (2016).

[12] Koch, A., Walzer, S. and Härtel, K.: Card-Based Cryptographic Protocols Using a Minimal Number of Cards, *Advances in Cryptology—ASIACRYPT 2015* (Iwata, T. and Cheon, J. H., eds.), LNCS, Vol. 9452, Berlin, Heidelberg, Springer, pp. 783–807 (online), available from ⟨https://doi.org/10.1007/978-3-662-48797-6_32⟩ (2015).

Table 7 Summary of Contradictions in Each of the Equations

Assumption	Operation *a*	Operation *b*	Operation *c*
$a_n + b_n + d_n + e_n = g_n$	$i_{n-1} = -a_{n-1} - d_{n-1}$	$h_{n-1} = -b_{n-1} - e_{n-1}$	$a_{n-1} + b_{n-1} + d_{n-1} + e_{n-1} = g_{n-1}$
$c_n + f_n = h_n + i_n$	$i_{n-1} = -a_{n-1} - d_{n-1}$	$h_{n-1} = -b_{n-1} - e_{n-1}$	$c_{n-1} + f_{n-1} = h_{n-1} + i_{n-1}$
$a_n + c_n + d_n + f_n = h_n$	$a_{n-1} + d_{n-1} = -i_{n-1}$	$a_{n-1} + c_{n-1} + d_{n-1} + f_{n-1} = h_{n-1}$	$c_{n-1} + f_{n-1} = -g_{n-1}$
$b_n + e_n = g_n + i_n$	$a_{n-1} + d_{n-1} = -i_{n-1}$	$b_{n-1} + e_{n-1} = g_{n-1} + i_{n-1}$	$c_{n-1} + f_{n-1} = -g_{n-1}$
$b_n + c_n + e_n + f_n = i_n$	$b_{n-1} + c_{n-1} + e_{n-1} + f_{n-1} = -i_{n-1}$	$h_{n-1} = -b_{n-1} - e_{n-1}$	$g_{n-1} = -c_{n-1} - f_{n-1}$
$a_n + d_n = g_n + i_n$	$a_{n-1} + d_{n-1} = g_{n-1} + i_{n-1}$	$h_{n-1} = -b_{n-1} - e_{n-1}$	$g_{n-1} = -c_{n-1} - f_{n-1}$

Table 8 state μ_{13}

♡♣♡♣	$1/4(X_{00} + X_{11})$
♣♣♡♡	$1/4(X_{00} + X_{11})$
♣♡♡♣	$1/2(X_{00} + X_{11})$
♡♣♣♡	$1/4(X_{01} + X_{10})$
♣♡♣♡	$3/8(X_{01} + X_{10})$
♡♡♣♣	$3/8(X_{01} + X_{10})$

Table 9 state μ_{14}

♡♣♡♣	$1/2(X_{00} + X_{11})$
♣♣♡♡	$1/4(X_{00} + X_{11})$
♣♡♡♣	$1/4(X_{00} + X_{11})$
♡♣♣♡	$3/8(X_{01} + X_{10})$
♣♡♣♡	$1/4(X_{01} + X_{10})$
♡♡♣♣	$3/8(X_{01} + X_{10})$

Table 10 state μ_{15}

♡♣♡♣	$1/4(X_{00} + X_{11})$
♣♣♡♡	$1/2(X_{00} + X_{11})$
♣♡♡♣	$1/4(X_{00} + X_{11})$
♡♣♣♡	$3/8(X_{01} + X_{10})$
♣♡♣♡	$3/8(X_{01} + X_{10})$
♡♡♣♣	$1/4(X_{01} + X_{10})$

Table 11 state μ_n

♡♣♡♣	$a_n(X_{00} + X_{11})$
♣♣♡♡	$b_n(X_{00} + X_{11})$
♣♡♡♣	$c_n(X_{00} + X_{11})$
♡♣♣♡	$d_n(X_{01} + X_{10})$
♣♡♣♡	$e_n(X_{01} + X_{10})$
♡♡♣♣	$f_n(X_{01} + X_{10})$

Table 12 state μ^a_{n+1}

♡♣♡♣	$(1/2)(a_n + c_n)(X_{00} + X_{11})$
♣♣♡♡	$b_n(X_{00} + X_{11})$
♣♡♡♣	$(1/2)(a_n + c_n)(X_{00} + X_{11})$
♡♣♣♡	$(1/2)(d_n + e_n)(X_{01} + X_{10})$
♣♡♣♡	$(1/2)(d_n + e_n)(X_{01} + X_{10})$
♡♡♣♣	$f_n(X_{01} + X_{10})$

Table 13 state μ^b_{n+1}

♡♣♡♣	$a_n(X_{00} + X_{11})$
♣♣♡♡	$(1/2)(b_n + c_n)(X_{00} + X_{11})$
♣♡♡♣	$(1/2)(b_n + c_n)(X_{00} + X_{11})$
♡♣♣♡	$(1/2)(d_n + f_n)(X_{01} + X_{10})$
♣♡♣♡	$e_n(X_{01} + X_{10})$
♡♡♣♣	$(1/2)(d_n + f_n)(X_{01} + X_{10})$

Table 14 state μ^c_{n+1}

♡♣♡♣	$1/3(a_n + b_n + c_n)(X_{00} + X_{11})$
♣♣♡♡	$1/3(a_n + b_n + c_n)(X_{00} + X_{11})$
♣♡♡♣	$1/3(a_n + b_n + c_n)(X_{00} + X_{11})$
♡♣♣♡	$1/3(d_n + e_n + f_n)(X_{01} + X_{10})$
♣♡♣♡	$1/3(d_n + e_n + f_n)(X_{01} + X_{10})$
♡♡♣♣	$1/3(d_n + e_n + f_n)(X_{01} + X_{10})$

Table 15 state μ_x

♡♣♡♡♣	$1/4(X_{00} + X_{01} + X_{10})$
♡♣♡♣♡	$1/8(X_{00} + X_{01} + X_{10})$
♣♣♡♡♡	$1/8(X_{00} + X_{01} + X_{10})$
♡♡♣♡♣	$1/4(X_{00} + X_{01} + X_{10})$
♡♡♣♣♡	$1/8(X_{00} + X_{01} + X_{10})$
♣♡♣♡♡	$1/8(X_{00} + X_{01} + X_{10})$
♡♡♡♣♣	$3/8(X_{11})$
♣♡♡♡♣	$3/8(X_{11})$
♣♡♡♣♡	$1/4(X_{11})$

Table 16 state μ_y

♡♣♡♡♣	$1/8(X_{00} + X_{01} + X_{10})$
♡♣♡♣♡	$1/8(X_{00} + X_{01} + X_{10})$
♣♣♡♡♡	$1/4(X_{00} + X_{01} + X_{10})$
♡♡♣♡♣	$1/8(X_{00} + X_{01} + X_{10})$
♡♡♣♣♡	$1/8(X_{00} + X_{01} + X_{10})$
♣♡♣♡♡	$1/4(X_{00} + X_{01} + X_{10})$
♡♡♡♣♣	$1/4(X_{11})$
♣♡♡♡♣	$3/8(X_{11})$
♣♡♡♣♡	$3/8(X_{11})$

Table 17 state μ_z

♡♣♡♡♣	$1/8(X_{00} + X_{01} + X_{10})$
♡♣♡♣♡	$1/4(X_{00} + X_{01} + X_{10})$
♣♣♡♡♡	$1/4(X_{00} + X_{01} + X_{10})$
♡♡♣♡♣	$1/8(X_{00} + X_{01} + X_{10})$
♡♡♣♣♡	$1/4(X_{00} + X_{01} + X_{10})$
♣♡♣♡♡	$1/8(X_{00} + X_{01} + X_{10})$
♡♡♡♣♣	$3/8(X_{11})$
♣♡♡♡♣	$1/4(X_{11})$
♣♡♡♣♡	$3/8(X_{11})$

Table 18 state μ_n

♡♣♡♡♣	$a_n(X_{00} + X_{01} + X_{10})$
♡♣♡♣♡	$b_n(X_{00} + X_{01} + X_{10})$
♣♣♡♡♡	$c_n(X_{00} + X_{01} + X_{10})$
♡♡♣♡♣	$d_n(X_{00} + X_{01} + X_{10})$
♡♡♣♣♡	$e_n(X_{00} + X_{01} + X_{10})$
♣♡♣♡♡	$f_n(X_{00} + X_{01} + X_{10})$
♡♡♡♣♣	g_nX_{11}
♣♡♡♡♣	h_nX_{11}
♣♡♡♣♡	i_nX_{11}

Table 19 state μ^a_{n+1}

♡♣♡♡♣	$a_n(X_{00} + X_{01} + X_{10})$
♡♣♡♣♡	$1/2(b_n + c_n)(X_{00} + X_{01} + X_{10})$
♣♣♡♡♡	$1/2(b_n + c_n)(X_{00} + X_{01} + X_{10})$
♡♡♣♡♣	$d_n(X_{00} + X_{01} + X_{10})$
♡♡♣♣♡	$1/2(e_n + f_n)(X_{00} + X_{01} + X_{10})$
♣♡♣♡♡	$1/2(e_n + f_n)(X_{00} + X_{01} + X_{10})$
♡♡♡♣♣	$1/2(g_n + h_n)X_{11}$
♣♡♡♡♣	$1/2(g_n + h_n)X_{11}$
♣♡♡♣♡	i_nX_{11}

Table 20 state μ^b_{n+1}

♡♣♡♡♣	$1/2(a_n + c_n)(X_{00} + X_{01} + X_{10})$
♡♣♡♣♡	$b_n(X_{00} + X_{01} + X_{10})$
♣♣♡♡♡	$1/2(a_n + c_n)(X_{00} + X_{01} + X_{10})$
♡♡♣♡♣	$1/2(d_n + f_n)(X_{00} + X_{01} + X_{10})$
♡♡♣♣♡	$e_n(X_{00} + X_{01} + X_{10})$
♣♡♣♡♡	$1/2(d_n + f_n)(X_{00} + X_{01} + X_{10})$
♡♡♡♣♣	$1/2(g_n + i_n)X_{11}$
♣♡♡♡♣	h_nX_{11}
♣♡♡♣♡	$1/2(g_n + i_n)X_{11}$

Table 21 state μ^c_{n+1}

♡♣♡♡♣	$1/2(a_n + b_n)(X_{00} + X_{01} + X_{10})$
♡♣♡♣♡	$1/2(a_n + b_n)(X_{00} + X_{01} + X_{10})$
♣♣♡♡♡	$c_n(X_{00} + X_{01} + X_{10})$
♡♡♣♡♣	$1/2(d_n + e_n)(X_{00} + X_{01} + X_{10})$
♡♡♣♣♡	$1/2(d_n + e_n)(X_{00} + X_{01} + X_{10})$
♣♡♣♡♡	$f_n(X_{00} + X_{01} + X_{10})$
♡♡♡♣♣	g_nX_{11}
♣♡♡♡♣	$1/2(h_n + i_n)X_{11}$
♣♡♡♣♡	$1/2(h_n + i_n)X_{11}$