# Six-Card Finite-Runtime XOR Protocol with Only Random Cut

Kodai Toyoda
Graduate School of Information Sciences, Tohoku University
Sendai, Japan

Daiki Miyahara
Graduate School of Information Sciences, Tohoku University
National Institute of Advanced Industrial Science and Technology
Sendai, Japan

Takaaki Mizuki
Cyberscience Center, Tohoku University
Sendai, Japan

Hideaki Sone
Cyberscience Center, Tohoku University
Sendai, Japan

## ABSTRACT

Executing a card-based cryptographic protocol is an attractive way to perform secure multiparty computation (MPC) with a deck of physical cards. Crèpeau and Kilian at CRYPTO 1993 proposed card-based AND and XOR protocols that can deal with a logical conjunction and exclusive-or of variables. Their protocols use a familiar shuffling action called a random cut that can be easily implemented by humans, while the numbers of required cards and shuffles are not small enough to be efficient and the protocols are Las Vegas algorithms, i.e., they are not finite-runtime. Several researchers have improved upon the protocols for a quarter of a century. Eventually, there are many AND protocols in the literature, some of which are quite efficient and practical to execute. By contrast, there are only three XOR protocols including the Crèpeau–Kilian protocol mentioned above. In this paper, we design an efficient XOR protocol using only random cuts; the numbers of required cards and shuffles are six and two (which is fixed), respectively. Our proposed XOR protocol is the first construction of a finite-runtime XOR protocol if we restrict ourselves to use only random cuts.

## CCS CONCEPTS

• **Security and privacy** → **Cryptography**.

## KEYWORDS

Secure multiparty computation; Card-based cryptography; Deck of cards

**ACM Reference Format:**
Kodai Toyoda, Daiki Miyahara, Takaaki Mizuki, and Hideaki Sone. 2020. Six-Card Finite-Runtime XOR Protocol with Only Random Cut. In *7th ACM ASIA Public-Key Cryptography Workshop (APKC'20), October 6, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3384940.3388961

## 1 INTRODUCTION

Executing a *card-based cryptographic protocol* is an attractive way to perform secure multiparty computation (MPC) with a deck of physical cards. Since den Boer [3] produced the "five-card trick" in 1989, many card-based protocols have been invented.



### 1.1 Card-Based AND and XOR Protocols

Most card-based protocols use a two-colored deck of cards, namely red $\boxed{\heartsuit}$ and black $\boxed{\clubsuit}$ cards having identical backs $\boxed{?}$. To manipulate Boolean values, we use the following encoding rule:

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

According to the above encoding rule, a player having their private bit $x \in \{0, 1\}$ can commit $x$ by putting two face-down cards on a table:

$$\underbrace{\boxed{?}\boxed{?}}_{x}.$$

We call such a pair of face-down cards a *commitment to* $x$.

Given two commitments to $a, b \in \{0, 1\}$ (along with six helping cards[1]), the Crèpeau–Kilian AND protocol [2] outputs a commitment to $a \wedge b$ without revealing any information about the values of $a$ and $b$:

$$\underbrace{\boxed{?}\boxed{?}}_{a}\ \underbrace{\boxed{?}\boxed{?}}_{b}\ \boxed{\clubsuit}\boxed{\heartsuit}\boxed{\spadesuit}\boxed{\spadesuit}\boxed{\diamondsuit}\boxed{\diamondsuit} \to \cdots \to \underbrace{\boxed{?}\boxed{?}}_{a\wedge b}.$$

Since the output value is derived as a commitment, it can be used as an input commitment for another computation. The drawback of the protocol is that the numbers of required cards and shuffles are not small enough to be efficient and it is a Las Vegas algorithm, i.e., the number of required shuffles is only expectedly finite. Several researchers have improved upon the protocol for a quarter of a century. Eventually, there are many AND protocols as shown in Table 1, some of which are quite efficient and practical to execute.

On the other hand, another important elementary computation is XOR (exclusive-or). As for XOR, Crèpeau and Kilian [2] also proposed an XOR protocol with 14 cards by modifying their AND protocol:

$$\underbrace{\boxed{?}\boxed{?}}_{a}\ \underbrace{\boxed{?}\boxed{?}}_{b}\ \boxed{\clubsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\spadesuit}\boxed{\spadesuit}\boxed{\diamondsuit}\boxed{\diamondsuit} \to \cdots \to \underbrace{\boxed{?}\boxed{?}}_{a\oplus b}.$$

[1] The Crèpeau–Kilian AND protocol uses a four-colored deck of cards.

**Table 1: The existing card-based AND protocols, where "RC" means the random cut and "RBC" means the random bisection cut.**

| | #Colors | #Cards | #Shuf. | Runtime | Type of shuf. |
|---|---|---|---|---|---|
| Crépeau–Kilian [2] | 4 | 10 | 8 (exp.) | Las Vegas | RC |
| Niemi–Renvall [11] | 2 | 12 | 7.5 (exp.) | Las Vegas | RC |
| Stiglic [12] | 2 | 8 | 2 (exp.) | Las Vegas | RC |
| Mizuki–Sone [9] | 2 | 6 | 1 | finite | RBC |
| Abe *et al.* [1] | 2 | 5 | 7 (exp.) | Las Vegas | RC & RBC |
| Five-card KWH [6] | 2 | 5 | 14/3 (exp.) | finite | not practical |
| Four-card KWH [6] | 2 | 4 | 8 (exp.) | Las Vegas | not practical |

That is, it uses 14 cards of four colors and ten shuffles on average. The shuffle their protocol uses is a familiar shuffling action called a *random cut* (which will be explained in Section 2).

To reduce the number of required cards and shuffles, Mizuki, Uchiike, and Sone [10] in 2006 proposed a Las Vegas XOR protocol with ten cards, improving upon the Crèpeau–Kilian protocol:


That is, it uses ten cards of two colors and seven shuffles (random cuts) on average. We will briefly introduce the protocol in Section 2.

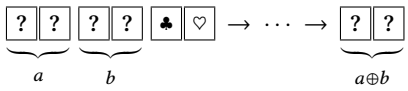
Three years after that, Mizuki and Sone [9] in 2009 proposed an efficient finite-runtime XOR protocol that requires only four cards and one shuffle.

Thus, in contrast to AND protocols (as shown in Table 1), there are only three XOR protocols including the Crèpeau–Kilian protocol as shown in Table 2.[2]

## 1.2 Contribution

See Table 2 again. The Mizuki–Sone XOR protocols [9] is the most efficient one with respect to the numbers of required cards and shuffles. Their protocol uses a shuffling action called a *random bisection cut* (not a random cut) that bisects a sequence of cards and then shuffles the two halves. A random bisection cut can be implemented securely by using additional tools [13]. Therefore, the random cut is better than the random bisection cut in terms of easy implementation. However, we note that there exists no finite-runtime XOR protocol using only random cuts, as known from Table 2. This means that showing the possibility or impossibility for finding such a protocol remains a challenging task in the research area of card-based protocols. It should be noted that in 2017 Koch and Walzer [5] proposed actively secure protocols depending only on random cuts and they implied that finding new protocols using only random cuts is an interesting open challenge.

In this paper, we affirmatively answer the above question. That is, we give a finite-runtime XOR protocol using only random cuts. See the bottom row in Table 2. Our proposed protocol requires six cards (i.e., two input commitments, a helping black card, and a red card):




The number of required shuffles (namely, random cuts) is exactly two (which is fixed).

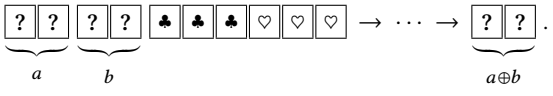
We will prove the correctness and security of our proposed six-card XOR protocol by using the *KWH-tree* (whose idea and notion were invented in [6]), from which we can visually confirm state transitions of the protocol.

## 1.3 Outline

The remainder of this paper is organized as follows. In Section 2, we introduce the Mizuki–Uchiike–Sone XOR protocol [10] to present what a card-based protocol is and what a random cut is. In Section 3, we present our six-card finite-runtime XOR protocol. In Section 4, we prove the correctness and security of our proposed protocol by showing its KWH-tree [6]. In Section 5, we mention the idea behind our proposed protocol. We conclude this paper in Section 6.

# 2 MIZUKI–UCHIIKE–SONE XOR PROTOCOL

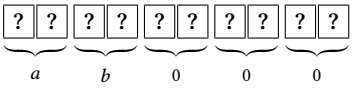
In this section, we introduce the Mizuki–Uchiike–Sone XOR protocol [10]; given two commitments to $a, b \in \{0, 1\}$ along with six helping cards (three black and three red cards), it outputs a commitment to $a \oplus b$:




The protocol proceeds as follows.

(1) Put the ten cards as follows:


Then, turn over the face-up cards:

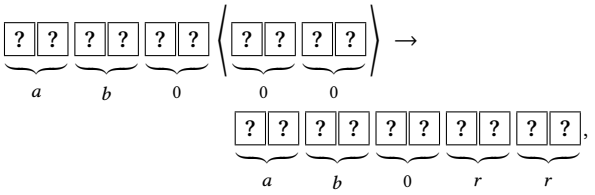



[2] In 2015 Koch, Walzer, and Härtel [6] presented a general protocol for any $n$-ary Boolean function with $2n$ cards. This implies the existence of a secure XOR protocol without any additional card using a non-practical shuffle, namely $(\mathsf{shuf}, \{(2\,3), (1\,2), (3\,4), (1\,3\,4\,2)\})$, although it needs to restart with a probability of 3/4.

**Table 2: The existing card-based XOR protocols and our proposed protocol.**

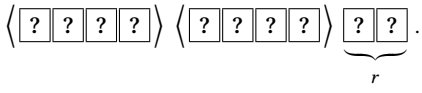
| | #Colors | #Cards | #Shuf. | Runtime | Type of shuf. |
|---|---|---|---|---|---|
| Crépeau–Kilian [2] | 4 | 14 | 10 (exp.) | Las Vegas | RC |
| Mizuki–Uchiike–Sone [10] | 2 | 10 | 7 (exp.) | Las Vegas | RC |
| Mizuki–Sone [9] | 2 | 4 | 1 | finite | RBC |
| Ours | 2 | 6 | 2 | finite | RC |

(2) Apply a *random cut* (denoted by $\langle \cdots \rangle$) to the rightmost four cards:

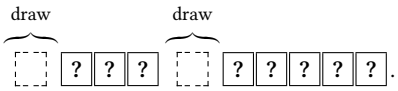



where $r$ is a random bit. A random cut is to cyclically shifts the sequence of cards at random without changing its order. Note that a random cut can be easily implemented with human hands so that nobody knows the offset [14].

(3) Apply two random cuts as follows:

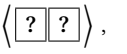



(4) Draw the first and fifth cards without revealing them:




(2)

Then, apply a random cut to the two drawn cards:

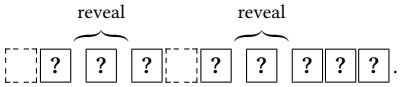


and reveal them.

(a) If the resulting sequence is ♣ ♡ or ♣ ♡, go to Step 5.

(b) If the resulting sequence is ♣ ♣ or ♡ ♡, return the two cards to the sequence (2) with their faces down. Then, go back to Step 3.

(5) Reveal the third and seventh cards:




(a) If the resulting sequence is ♣ ♡ or ♣ ♡ (which implies $a \oplus b \oplus 0 \oplus r\ (= a \oplus b \oplus r) = 0$), the rightmost two cards (i.e., the commitment to $r$) form the output $a \oplus b$.

(b) If the resulting sequence is ♣ ♣ or ♡ ♡, (which implies $a \oplus b \oplus 0 \oplus r = 1$), the rightmost two cards form $\overline{a \oplus b}$. Then, reverse the order of the two cards consisting of the commitment to $\overline{a \oplus b}$ to obtain a commitment to $a \oplus b$.

The numbers of required cards and shuffles for this XOR protocol are ten and seven (which is an expected number), respectively; this is a Las Vegas protocol because the protocol (in Step 4) has the possibility of going back to the previous step (with a probability of 1/2). See Table 2 again.

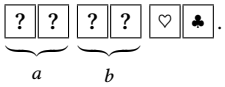
## 3 OUR XOR PROTOCOL

In this section, we construct a six-card finite-runtime XOR protocol using only random cuts.

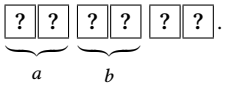
### 3.1 Description

Here, we give the description of our protocol.

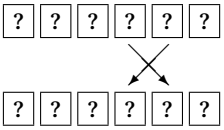
(1) Put the two input commitments and the two helping cards as follows:



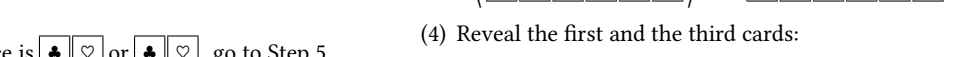

Then, turn over the two face-up cards:



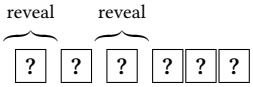

(2) Rearrange the order of the sequence as follows:



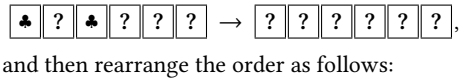
(3) Apply a random cut to the sequence of six cards:



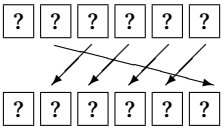
(4) Reveal the first and the third cards:



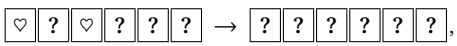

(a) If ♣ ♣ appears, then turn them face down:



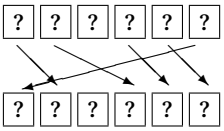
and then rearrange the order as follows:



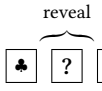
(b) If ♡ ♡ appears, then turn them face down:



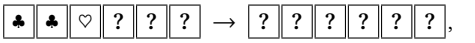
and then rearrange the order as follows:



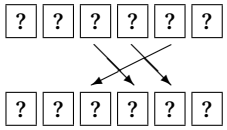
(c) If ♣ ♡ appears, then reveal the second card:



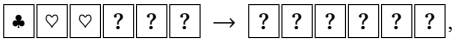

(i) If ♣ appears, then turn over the three face-up cards:



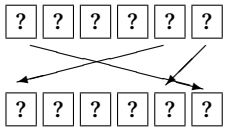
and then rearrange the order as follows:



(ii) If ♡ appears, then turn over the three face-up cards:

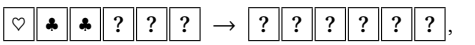


and then rearrange the order as follows:

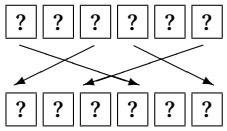


(d) If ♡ ♣ appears, then reveal the second card.

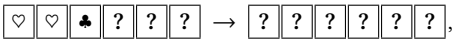
(i) If ♣ appears, then turn over the three face-up cards:



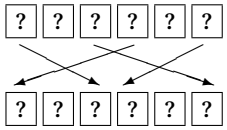
and then rearrange the order as follows:



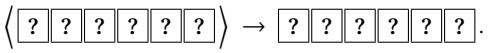
(ii) If ♡ appears, then turn over the three face-up cards:



and then rearrange the order as follows:

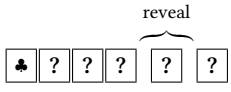


(5) Apply a random cut to the sequence of six cards:

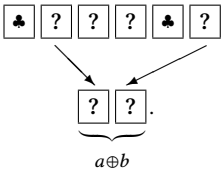


(6) Reveal the first card.

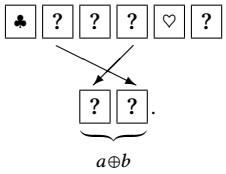
(a) If ♣ appears, then reveal the fifth card:




(i) If ♣ appears, then the second and sixth cards form a commitment to $a \oplus b$:

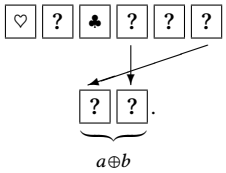



(ii) If ♡ appears, then the fourth and second cards form a commitment to $a \oplus b$:

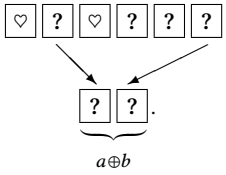



(b) If ♡ appears, then reveal the third card.

(i) If ♣ appears, then the sixth and fourth cards form a commitment to $a \oplus b$:




(ii) If ♡ appears, then the second and sixth cards form a commitment to $a \oplus b$:




This is our XOR protocol. Since this protocol has no loop, it terminates within a fixed number of shuffles (namely, random cuts), which is exactly two. See the bottom row in Table 2 again.

## 3.2 Pseudocode

The following is a pseudocode for our protocol, where we define

$$\mathsf{RC}_6 \stackrel{\text{def}}{=} \{\mathsf{id},\ (1\,2\,3\,4\,5\,6),\ (1\,2\,3\,4\,5\,6)^2,\ (1\,2\,3\,4\,5\,6)^3,\ (1\,2\,3\,4\,5\,6)^4,\ (1\,2\,3\,4\,5\,6)^5\},$$

$(\mathsf{perm}, \pi)$ for a permutation $\pi$ means to permute the sequence of cards according to $\pi$, $(\mathsf{turn}, T)$ for a set of positions of cards $T$ means to turn over all cards whose positions are in $T$, $(\mathsf{shuf}, \Pi)$ for a set of permutations $\Pi$ means to apply $(\mathsf{perm}, \pi)$ such that $\pi$ is drawn from $\Pi$ uniformly, and $(\mathsf{result}, i, j)$ specifies output positions.

---

input set:

$$\left\{\left(\tfrac{?}{\heartsuit}, \tfrac{?}{\clubsuit}, \tfrac{?}{\heartsuit}, \tfrac{?}{\clubsuit}, \tfrac{?}{\heartsuit}, \tfrac{?}{\clubsuit}\right), \left(\tfrac{?}{\heartsuit}, \tfrac{?}{\clubsuit}, \tfrac{?}{\clubsuit}, \tfrac{?}{\heartsuit}, \tfrac{?}{\heartsuit}, \tfrac{?}{\clubsuit}\right), \left(\tfrac{?}{\clubsuit}, \tfrac{?}{\heartsuit}, \tfrac{?}{\heartsuit}, \tfrac{?}{\clubsuit}, \tfrac{?}{\heartsuit}, \tfrac{?}{\clubsuit}\right), \left(\tfrac{?}{\clubsuit}, \tfrac{?}{\heartsuit}, \tfrac{?}{\clubsuit}, \tfrac{?}{\heartsuit}, \tfrac{?}{\heartsuit}, \tfrac{?}{\clubsuit}\right)\right\}$$

(perm, (4 5))
(shuf, $\mathsf{RC}_6$)
(turn, {1, 3})
**if** visible seq. = (♣, ?, ♣, ?, ?, ?) **then**
(turn, {1, 3})
(perm, (2 6 5 4 3))
**else if** visible seq. = (♡, ?, ♡, ?, ?, ?) **then**
(turn, {1, 3})

```
    (perm, (1 2 4 5 6))
  else if visible seq. = (♣, ?, ♡, ?, ?, ?) then
    (turn, {2})
    if visible seq. = (♣, ♣, ♡, ?, ?, ?) then
      (turn, {1, 2, 3})
      (perm, (3 4 5))
    else if visible seq. = (♣, ♡, ♡, ?, ?, ?) then
      (turn, {1, 2, 3})
      (perm, (1 6 5))
  else if visible seq. = (♡, ?, ♣, ?, ?, ?) then
    (turn, {2})
    if visible seq. = (♡, ♣, ♣, ?, ?, ?) then
      (turn, {1, 2, 3})
      (perm, (1 4 6 3))
    else if visible seq. = (♡, ♡, ♣, ?, ?, ?) then
      (turn, {1, 2, 3})
      (perm, (1 3 6 4))
  (shuf, RC_6)
  (turn, {1})
  if visible seq. = (♣, ?, ?, ?, ?, ?) then
    (turn, {5})
    if visible seq. = (♣, ?, ?, ?, ♣, ?) then
      (result, 2, 6)
    else if visible seq. = (♣, ?, ?, ?, ♡, ?) then
      (result, 4, 2)
  else if visible seq. = (♡, ?, ?, ?, ?, ?) then
    (turn, {3})
    if visible seq. = (♡, ?, ♣, ?, ?, ?) then
      (result, 6, 4)
    else if visible seq. = (♡, ?, ♡, ?, ?, ?) then
      (result, 2, 6)
```

In the next section, we confirm that our protocol definitively produces a commitment to $a \oplus b$ without leaking any information about $a$ and $b$.

## 4 CORRECTNESS AND SECURITY

In this section, we verify the correctness and security of the protocol presented in Section 3.

An XOR protocol is said to be *correct* if, given input commitments to $x, y$, it always produces a commitment to $x \oplus y$. We call a protocol is *secure* if it leaks no information for any run of the protocol (in other words, random variables $I$ and $V$ denoting the inputs and the visible sequence trace, respectively, are stochastically independent, where the visible sequence trace means what can be observed on the table). See [6–8] for the more formal definitions based on abstract machine and information theory.

To confirm that our protocol is correct and secure, we make use of the *KWH-tree*, which is a beautiful tool developed by Koch, Walzer, and Härtel [6]. That is, if one is able to write a KWH-tree satisfying some properties for a protocol, then it automatically implies that the protocol is correct and secure; see [4, 6, 8] for the details.

We describe the KWH-tree of our proposed protocol in Figure 1. In this figure, we call each box a *state*. The first state (box) in Figure 1 corresponds to an initial sequence, consisting of two input commitments and two helping cards; $X_{00}$, $X_{01}$, $X_{10}$, and $X_{11}$ represent the probabilities of

$$(a, b) = (0, 0),\ (a, b) = (0, 1),\ (a, b) = (1, 0), \text{ and } (a, b) = (1, 1),$$

respectively. A polynomial annotating a card sequence in a state, such as $1/6X_{00}$, represents the conditional probability that the current sequence is the one next to the polynomial, given the visible sequence trace observed so far on the table. From the four states (boxes) at the bottom, one can see that a commitment to $a \oplus b$ is definitively obtained. Furthermore, in each state, the sum of all polynomials is equal to $X_{00} + X_{01} + X_{10} + X_{11}$, implying that no information about $a$ and $b$ leaks, i.e., the inputs and visible sequence trace are stochastically independent. It should be noted that when a turn action is applied, for each possible visible sequence, there is a constant $c$ such that its probability is $c\ (X_{00} + X_{01} + X_{10} + X_{11})$.

Thus, the KWH-tree in Figure 1 guarantees that our proposed protocol is correct and secure. (As stated in Proposition 1 of [4], the KWH-tree is a witness for the correctness and security.)

**Table 3: Possible sequences of six cards divided into four equivalent classes, based on cyclic rotations.**

| (A) | (B) | (C) | (D) |
|---|---|---|---|
| (♡♣♡♣♡♣) | (♡♡♡♣♣♣) | (♡♡♣♡♣♣) | (♡♡♣♣♡♣) |
| (♣♡♣♡♣♡) | (♡♡♣♣♣♡) | (♡♣♡♣♣♡) | (♡♣♣♡♣♡) |
| | (♡♣♣♣♡♡) | (♣♡♣♣♡♡) | (♣♣♡♣♡♡) |
| | (♣♣♣♡♡♡) | (♡♣♣♡♡♣) | (♣♡♣♡♡♣) |
| | (♣♣♡♡♡♣) | (♣♣♡♡♣♡) | (♡♣♡♡♣♣) |
| | (♣♡♡♡♣♣) | (♣♡♡♣♡♣) | (♣♡♡♣♣♡) |

## 5 IDEA BEHIND OUR PROTOCOL

In this section, we explain the idea behind our protocol.

As seen in Figure 1, the initial state of our protocol can be written as

$$\{(♡♣♡♣♡♣, X_{11}), (♡♣♣♡♡♣, X_{10}),$$
$$(♣♡♡♣♡♣, X_{01}), (♣♡♣♡♡♣, X_{00})\}.$$

To generate a commitment to $a \oplus b$, we want to "mix" $X_{11}$ and $X_{00}$ (where the result is $1 \oplus 1 = 0 \oplus 0 = 0$) and mix $X_{10}$ and $X_{01}$ (where the result is $1 \oplus 0 = 0 \oplus 1 = 1$) so that we have the bottom-most states in Figure 1, such as

$$\{(♣♣♡♡♣♡, X_{11} + X_{00}), (♣♡♡♡♣♣, X_{10} + X_{01})\}.$$

Let us show how to achieve this, as below.

Note that there are 20 combinations of possible sequences of six cards (three red cards and three black cards). Considering that the random cut is a cyclic shuffling operation, let us divide these sequences into four equivalent classes, as shown in Table 3. When a random cut is applied to a sequence, the resulting sequence equally likely becomes one of the sequences that belong to the same class. For example, if we apply a random cut to a state

$$\{(♡♣♡♡♣♣, X_{11})\},$$ [3]

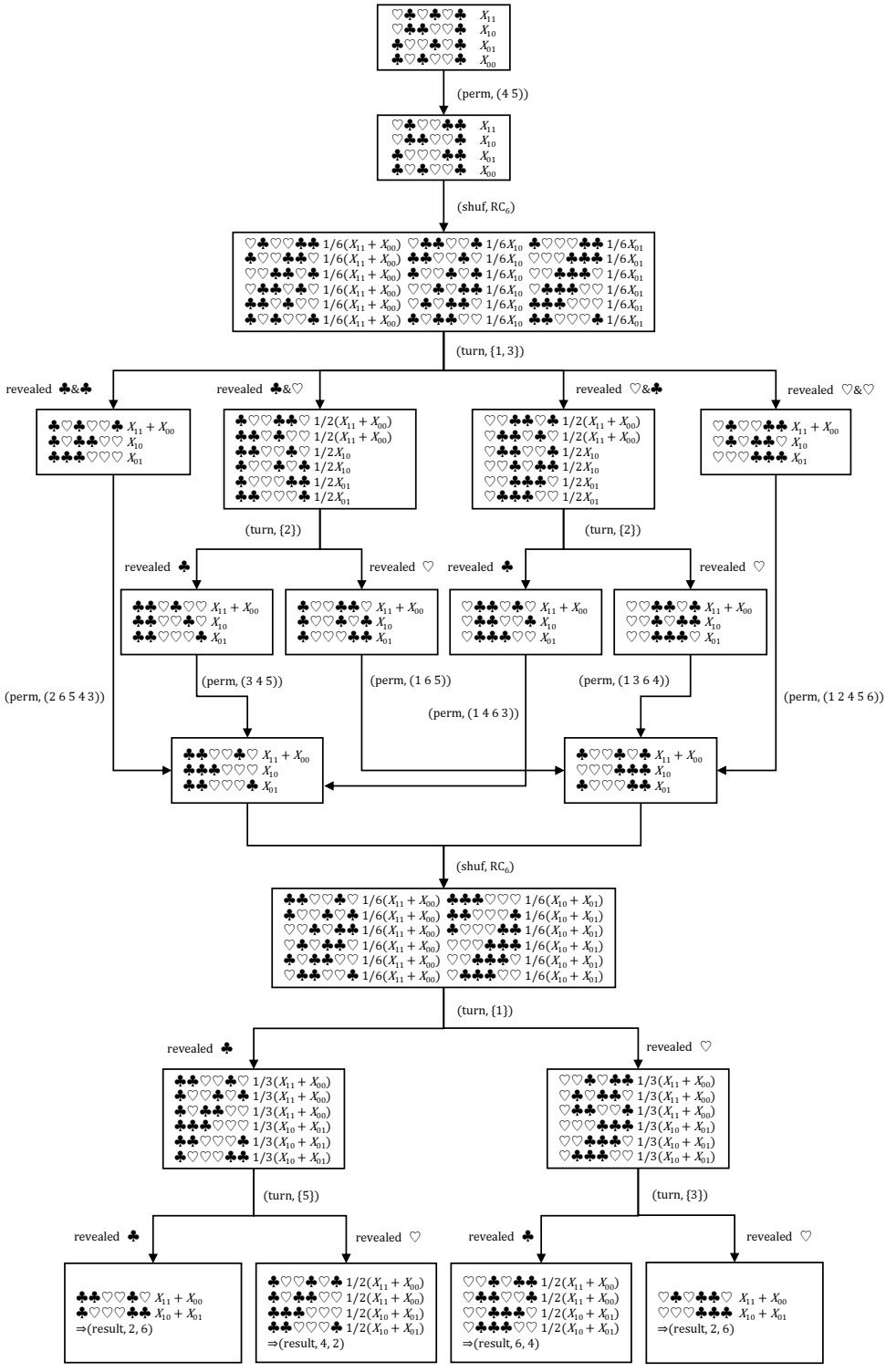



**Figure 1: The KWH-tree of our six-card finite-runtime XOR protocol.**

the resulting state will be

$$\{(\heartsuit\heartsuit\clubsuit\clubsuit\heartsuit\clubsuit, 1/6X_{11}), (\heartsuit\clubsuit\clubsuit\heartsuit\clubsuit\heartsuit, 1/6X_{11}),$$
$$(\clubsuit\clubsuit\heartsuit\clubsuit\heartsuit\heartsuit, 1/6X_{11}), (\clubsuit\heartsuit\clubsuit\heartsuit\heartsuit\clubsuit, 1/6X_{11}),$$
$$(\heartsuit\clubsuit\heartsuit\heartsuit\clubsuit\clubsuit, 1/6X_{11}), (\clubsuit\heartsuit\heartsuit\clubsuit\clubsuit\heartsuit, 1/6X_{11})\}.$$

For another example, if we apply a random cut to a state

$$\{(\heartsuit\clubsuit\heartsuit\heartsuit\clubsuit\clubsuit, X_{11}), (\clubsuit\heartsuit\clubsuit\heartsuit\heartsuit\clubsuit, X_{00})\},$$

the resulting state will be

$$\{(\heartsuit\heartsuit\clubsuit\clubsuit\heartsuit\clubsuit, 1/6(X_{11}+X_{00})), (\heartsuit\clubsuit\clubsuit\heartsuit\clubsuit\heartsuit, 1/6(X_{11}+X_{00})),$$
$$(\clubsuit\clubsuit\heartsuit\clubsuit\heartsuit\heartsuit, 1/6(X_{11}+X_{00})), (\clubsuit\heartsuit\clubsuit\heartsuit\heartsuit\clubsuit, 1/6(X_{11}+X_{00})),$$
$$(\heartsuit\clubsuit\heartsuit\heartsuit\clubsuit\clubsuit, 1/6(X_{11}+X_{00})), (\clubsuit\heartsuit\heartsuit\clubsuit\clubsuit\heartsuit, 1/6(X_{11}+X_{00}))\}.$$

As in this second example, we are able to mix $X_{11}$ and $X_{00}$ by a random cut, i.e., from the resulting sequence we cannot know whether the input is (0,0) or (1,1). Because just applying a random cut to the initial state

$$\{(\heartsuit\clubsuit\heartsuit\clubsuit\heartsuit\clubsuit, X_{11}), (\heartsuit\clubsuit\clubsuit\heartsuit\heartsuit\clubsuit, X_{10}),$$
$$(\clubsuit\heartsuit\heartsuit\clubsuit\heartsuit\clubsuit, X_{01}), (\clubsuit\heartsuit\clubsuit\heartsuit\heartsuit\clubsuit, X_{00})\}$$

cannot mix $X_{11}$ and $X_{00}$, we first apply a permutation (4 5) so that some of the resulting sequences after a random cut become cyclically equivalent. Thus, $X_{11}$ and $X_{00}$ are now mixed as shown in the third state from the top in Figure 1.

Next, we want to mix $X_{10}$ and $X_{01}$. For this, we first reveal some cards to "reduce" the number of possibilities. For example, if we reveal the first and third cards and they are both black, then the resulting state will be

$$\{(\clubsuit\heartsuit\clubsuit\heartsuit\heartsuit\clubsuit, X_{11}+X_{00}), (\clubsuit\heartsuit\clubsuit\clubsuit\heartsuit\heartsuit, X_{10}), (\clubsuit\clubsuit\clubsuit\heartsuit\heartsuit\heartsuit, X_{01})\}.$$

Then, it suffices to apply a permutation (2 6 5 4 3) to make the last two sequences belong to the same class (while keeping the first sequence in a distinct class). In other cases as well, by applying a permutation corresponding to each state and applying a random cut, we can mix $X_{10}$ and $X_{01}$ as shown in the third state from the bottom in Figure 1.

Finally, we turn over two cards to obtain a commitment to $a \oplus b$. For example, if we reveal the first and fifth cards and they are both black, then the resulting state will be

$$\{(\clubsuit\clubsuit\heartsuit\heartsuit\clubsuit\heartsuit, X_{11}+X_{00}), (\clubsuit\heartsuit\heartsuit\heartsuit\clubsuit\clubsuit, X_{10}+X_{01})\},$$

and the second and sixth cards constitute a commitment to $a \oplus b$.

## 6 CONCLUSION

In this paper, we constructed a six-card finite-runtime XOR protocol using only random cuts; it uses exactly only two random cuts. This is the first finite-runtime XOR protocol using only random cuts.

It is an interesting open problem whether there is an XOR protocol that uses only random cuts and less than six cards.

[3]Note that this example never appears in any secure XOR protocol.


## ACKNOWLEDGMENTS

We thank the anonymous referees, whose comments have helped us to improve the presentation of the paper. We would like to thank Yuta Abe for his helpful discussions and support. This work was supported by JSPS KAKENHI Grant Numbers JP17K00001 and JP19J21153.


## REFERENCES


[1] Yuta Abe, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. 2018. Five-Card AND Protocol in Committed Format Using Only Practical Shuffles. In *5th ACM on ASIA Public-Key Cryptography Workshop* (Incheon, Republic of Korea) *(APKC '18)*. Association for Computing Machinery, New York, NY, USA, 3–8. https://doi.org/10.1145/3197507.3197510

[2] Claude Crépeau and Joe Kilian. 1994. Discreet Solitary Games. In *Advances in Cryptology—CRYPTO' 93 (LNCS)*, Douglas R. Stinson (Ed.), Vol. 773. Springer, Berlin, Heidelberg, 319–330. https://doi.org/10.1007/3-540-48329-2_27

[3] Bert Den Boer. 1990. More Efficient Match-Making and Satisfiability The Five Card Trick. In *Advances in Cryptology—EUROCRYPT '89 (LNCS)*, Jean-Jacques Quisquater and Joos Vandewalle (Eds.), Vol. 434. Springer, Berlin, Heidelberg, 208–217. https://doi.org/10.1007/3-540-46885-4_23

[4] Julia Kastner, Alexander Koch, Stefan Walzer, Daiki Miyahara, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. 2017. The Minimum Number of Cards in Practical Card-Based Protocols. In *Advances in Cryptology—ASIACRYPT 2017 (LNCS)*, Tsuyoshi Takagi and Thomas Peyrin (Eds.), Vol. 10626. Springer, Cham, 126–155. https://doi.org/10.1007/978-3-319-70700-6_5

[5] Alexander Koch and Stefan Walzer. 2017. Foundations for Actively Secure Card-based Cryptography. Cryptology ePrint Archive, Report 2017/423. https://eprint.iacr.org/2017/423.

[6] Alexander Koch, Stefan Walzer, and Kevin Härtel. 2015. Card-Based Cryptographic Protocols Using a Minimal Number of Cards. In *Advances in Cryptology—ASIACRYPT 2015 (LNCS)*, Tetsu Iwata and Jung Hee Cheon (Eds.), Vol. 9452. Springer, Berlin, Heidelberg, 783–807. https://doi.org/10.1007/978-3-662-48797-6_32

[7] Takaaki Mizuki and Hiroki Shizuya. 2014. A formalization of card-based cryptographic protocols via abstract machine. *Int. J. Inf. Secur.* 13, 1 (2014), 15–23. https://doi.org/10.1007/s10207-013-0219-4

[8] Takaaki Mizuki and Hiroki Shizuya. 2017. Computational Model of Card-Based Cryptographic Protocols and Its Applications. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* E100.A, 1 (2017), 3–11. https://doi.org/10.1587/transfun.E100.A.3

[9] Takaaki Mizuki and Hideaki Sone. 2009. Six-Card Secure AND and Four-Card Secure XOR. In *Frontiers in Algorithmics (LNCS)*, Xiaotie Deng, John E. Hopcroft, and Jinyun Xue (Eds.), Vol. 5598. Springer, Berlin, Heidelberg, 358–369. https://doi.org/10.1007/978-3-642-02270-8_36

[10] Takaaki Mizuki, Fumishige Uchiike, and Hideaki Sone. 2006. Securely computing XOR with 10 cards. *The Australasian Journal of Combinatorics* 36 (2006), 279–293. https://ajc.maths.uq.edu.au/?page=get_volumes&volume=36

[11] Valtteri Niemi and Ari Renvall. 1998. Secure multiparty computations without computers. *Theor. Comput. Sci.* 191, 1–2 (1998), 173–183. https://doi.org/10.1016/S0304-3975(97)00107-2

[12] Anton Stiglic. 2001. Computations with a deck of cards. *Theor. Comput. Sci.* 259, 1–2 (2001), 671–678. https://doi.org/10.1016/S0304-3975(00)00409-6

[13] Itaru Ueda, Daiki Miyahara, Akihiro Nishimura, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. 2020. Secure implementations of a random bisection cut. *Int. J. Inf. Secur.* 19, 4 (2020), 445–452. https://doi.org/10.1007/s10207-019-00463-w

[14] Itaru Ueda, Akihiro Nishimura, Yu-ichi Hayashi, Takaaki Mizuki, and Hideaki Sone. 2016. How to Implement a Random Bisection Cut. In *Theory and Practice of Natural Computing (LNCS)*, Carlos Martín-Vide, Takaaki Mizuki, and Miguel A. Vega-Rodríguez (Eds.), Vol. 10071. Springer, Cham, 58–69. https://doi.org/10.1007/978-3-319-49001-4_5