Card-based secure evaluation of decision trees

Yoshifumi Manabe[1][0000−0002−6312−257X] and Naoki Kobayashi[1]

School of Informatics, Kogakuin University,
Shinjuku, Tokyo 163-8677 Japan. manabe@cc.kogakuin.ac.jp

Abstract. Decision trees such as BDD (Binary Decision Diagrams) are commonly used for decision-making. The structure of the decision tree might be the know-how of the owner of the tree. Thus, when Alice provides a decision tree and Bob evaluates the tree, Alice wants to hide the structure of the decision tree from Bob. Though Bob can know the information about the nodes Bob traversed, the other node and edge information must be hidden. During the evaluation, Bob might use his private information. Thus, the traversed nodes and edges and the final result need to be hidden from Alice. To achieve these requirements, we propose a card-based decision tree evaluation protocol. By the nature of card-based cryptographic protocols that they are executed in a public place, the protocol is simpler than the ones executed on computers.

Keywords: card-based cryptography · secure multi-party computation, · decision tree · BDD

1 Introduction

Card-based cryptographic protocols [9,10] were proposed in which physical cards are used instead of computers to securely compute values. They can be used when users cannot trust the software on the computer. Also, the protocols are easy to understand, thus the protocols can be used to teach the basics of cryptography [2, 8] to accelerate the social implementation of advanced cryptography [3]. den Boer [1] first showed a five-card protocol to securely compute the logical AND of two inputs. Since then, many protocols have been proposed to realize primitives to compute any Boolean functions [5, 6, 11, 14] and specific computations such as millionaires' problem [12, 13] and so on.

This paper proposes a new application protocol to securely evaluate decision trees. Decision trees such as BDD (Binary Decision Diagrams) are commonly used for decision-making. We can find many examples of decision trees in disease diagnosis, for example, in [16]. Secure evaluations of the decision tree was discussed when the cloud server has the decision tree and the server calculates the result without knowing the user's private input data [4, 15, 17].

The structure of the decision tree might be the know-how of the owner of the tree. Thus, when Alice provides a decision tree and Bob evaluates the tree, Alice wants to hide the structure of the decision tree from Bob. Though Bob can know the information about the nodes Bob traversed, the other node and edge

information must be hidden. During the evaluation, Bob might use his private information. Thus, the traversed nodes and edges and the final result need to be hidden from Alice. To achieve these requirements, we propose a card-based decision tree evaluation protocol. The protocols in [4,15,17] need encryption with advanced functionality such as homomorphic encryption since the server needs to calculate the result without knowing the user's private data. The protocols are therefore complicated. By the nature of card-based cryptographic protocols that they are executed in a public place, our protocol is simpler than the ones executed on computers.

Section 2 gives the definitions of decision trees and cards used in our protocol. Section 3 shows the preparation to add dummy nodes to the decision trees. Section 4 shows the case of a complete binary tree. Section 5 shows the case of general binary trees. Section 6 evaluates the execution times of the proposed protocols. Section 7 concludes the paper.

2 Preliminaries

2.1 Decision Trees

This section shows the definition of the decision trees discussed in this paper. Though this paper considers binary decision trees, the result in this paper can be easily applied to general decision trees whose outdegrees in all internal nodes are the same. Decision tree $D = (V = V_1 \cup V_2, E)$ is a directed acyclic graph that has a unique root node $r \in V_1$. V_1 is the set of internal nodes and V_2 is the set of leaf nodes. The root node has no incoming edge. Each leaf node has no outgoing edge. Each node $v \in V_1$ has two outgoing edges $y(v), n(v) \in E$. $y(v) = (v, w)$ for some node $w \in V$ and $n(v) = (v, w')$ for some node $w' \in V$. Every node is reachable from the root node r.

Each internal node $v \in V_1$ has a condition $c(v)$. Each leaf node $v \in V_2$ has the decision value $d(v)$. A user u of the decision tree executes the following evaluation. u first evaluates $c(r)$ in the root node r. If $c(r) =$"yes" ("no") for u, u moves to the next node connected by edge $y(r)(n(r))$, respectively. At the new node, v', u evaluates the condition $c(v')$ and moves using either $y(v')$ or $n(v')$. Since there is no directed cycle in D, user u eventually moves to a leaf node $v \in V_2$. u obtains the decision $d(v)$. Fig. 1 shows an example of D. When the decision tree is used for disease diagnosis, the decisions of the leaf nodes are the names of diseases and the conditions are "The blood pressure is 135/85mmHg or higher?", "The color of the nails is cloudy like in this photo?", and so on.

Note that this paper assumes Bob can evaluate $c(v)$ using Bob's private value. The evaluation method is out of the scope of the paper. If $c(v)$ is a function f using Bob's secure input value $b_i (i = 1, 2, \ldots, \alpha)$ and f must be hidden from Bob, a secure function evaluation method might be necessary.

2.2 Security of decision tree

The security definitions when Alice has a decision tree and Bob has private data for the evaluation are given as follows. The players are assumed to be semi-

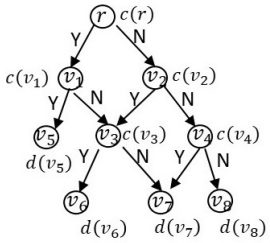

Fig. 1. An example of the decision tree.

honest, that is, they obey the rules of the protocols but try to obtain secure data.

Definition 1 (Security of Bob's private data).

A security game is defined between challenger B and an adversary A who has the decision tree T. A first gives two sequences of answers S_0, S_1, which is the list of evaluation results "yes"/"no" for each internal node to the leaf node. B privately selects random bit $b \in \{0,1\}$. A and B execute the decision tree evaluation protocol. B evaluates the conditions of internal nodes using S_b. After the evaluation, A outputs bit $b' \in \{0,1\}$. User data is secure if $|Pr[b = b'] - 1/2|$ is negligible.

The adversary has the tree and decides two different answers by Bob. B selects one of the two sequences of answers. A guesses which sequence is used. A wins if the probability of a correct guess is more than $1/2$. If such an adversary does not exist, the user data is kept secret.

Definition 2 (Security of Alice's decision tree).

A security game is defined between challenger A and an adversary B. B first outputs two decision trees T_0, T_1 that have a common path P from the root r to a leaf node v_0 and the depth of every leaf node is the same. Challenger A randomly selects $b \in \{0,1\}$. A and B executes the decision tree evaluation protocol with T_b and B selects the decision at each node to follow the path P. After the evaluation, B outputs $b' \in \{0,1\}$. The decision tree is secure if $|Pr[b = b'] - 1/2|$ is negligible.

The adversary gives two trees. One of the trees is used for the protocol execution. After the protocol execution, the adversary guesses which tree is used. B wins if the probability of a correct guess is more than $1/2$. If such an adversary does not exist, the decision tree is kept secret.

2.3 Definitions related to the cards

We show basic assumptions and operations for card-based cryptographic protocols. This paper uses $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$ cards. The cards with the same mark are

indistinguishable. The back of all cards is the same $\boxed{?}$. It is impossible to know the mark of face-down cards. In addition, there are cards to write the condition on each node and the final results. The cards are called condition cards. The back of all condition cards are the same $\boxed{?}$.

Set, turn, and shuffle operations are executed on cards. Setting cards is placing cards according to some rules. Turning a face-down card is opening a card and seeing the mark on the card. Turning a face-up card is hiding the mark of a card. Turing might be privately done. For example, Bob turns a face-down card, sees the mark, and turns the face-up card again at a place where Alice cannot see. Private operations can be done in the back or under the table.

Shuffle is the randomization of a sequence of cards. For a given sequence of face-down cards $c_1, c_2, \ldots, c_n$, shuffle returns a sequence $c_{\pi(1)}, c_{\pi(2)}, \ldots, c_{\pi(n)}$ for some permutation $\pi : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$. The permutation π is unknown to all players. Pile-scramble shuffle is a shuffle on piles of cards. A pile of cards P_i is a sequence of cards $c_{i,1}, c_{i,2}, \ldots, c_{i,m}$ whose order is fixed. Such a pile can be obtained by inserting the cards into an envelope. Then a shuffle is executed to the envelopes. After the shuffle, the cards in each envelope are set back. By a pile-scramble shuffle on piles $P_1, P_2, \ldots, P_n$, we obtain piles $P_{\pi(1)}, P_{\pi(2)}, \ldots, P_{\pi(n)}$. The sequence in each pile is unchanged by a pile-scramble shuffle, that is, if P_i is moved to the j-th pile by the pile-scramble shuffle, $c_{i,k}$ is moved to the k-th card in the j-th pile for all $k (1 \leq k \leq m)$. Note that the number of cards in every pile must be the same to hide the permutation π to the players.

3 Adding dummy edges and nodes to decision trees

For the preparation, Alice needs to set the depth of every leaf node is the same value. If the depth of some leaves is smaller, Bob knows some information about the structure of the subtree Bob evaluated. Since Bob does not want Alice to know the final decision, Bob does not want to let Alice know that Bob traversed to a small depth leaf node. Thus, in many cases, both Alice and Bob have the incentive to make the depths of all leaf nodes the same value. Alice adds some dummy edges and nodes and makes the depth of every leaf node the same value k. Fig. 2 shows an example of adding dummy nodes to the tree in Fig. 1. Node v'_1 is the dummy node and the condition $c(v'_1)$ is some dummy condition meaningless to the final decision. The algorithm to add dummy edges and nodes is out of the scope of the paper. In the rest of this paper, we assume that the depth of every leaf node is the same.

As shown in Fig. 2, the structure of the decision tree is not completely hidden by just making the depth of every leaf node the same. This paper discusses two methods for the secure evaluation. The first method is making copies of nodes. If an internal node has more than one incoming edge, Alice makes copies of such an internal node and makes the decision tree a complete binary tree as Fig. 3. An evaluation protocol for a complete binary tree is shown in Section 4.

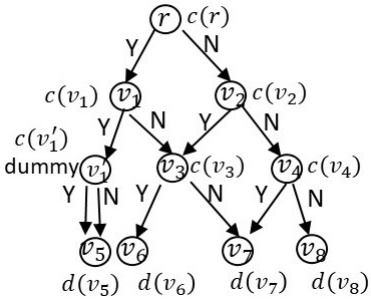

Fig. 2. Adding dummy nodes to modify the depth.

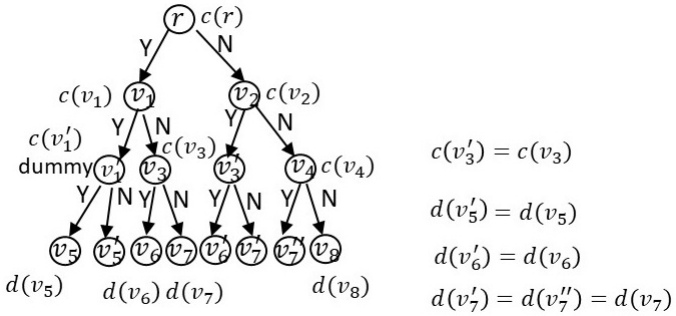

Fig. 3. Making a complete binary tree.

The method needs copying condition card $c(v)$. Such a copy might not be easy in some cases for example, $c(v)$ is testing with a color image on the card and the color is very hard to copy with a standard copy machine (for example, images for color blindness testing). Section 5 shows an evaluation protocol without copying internal nodes.

4 Protocol for complete binary tree

When the decision tree is a complete binary tree just like Fig. 3, randomization is simple. The condition card for node i is written as $c(i)$.

For each internal node, add a face-down $\boxed{\heartsuit}$ ($\boxed{\clubsuit}$) card to the "yes" ("no") edge, respectively, as shown in Fig. 4. Note that the cards are shown in face-up in Fig. 4 for explanation. Note that edges are unnecessary to set the binary tree since the outdegree of every internal node is two. The players place the nodes by the order of the breadth-first search. Then they can know the left and right child of each node from the sequence of nodes.

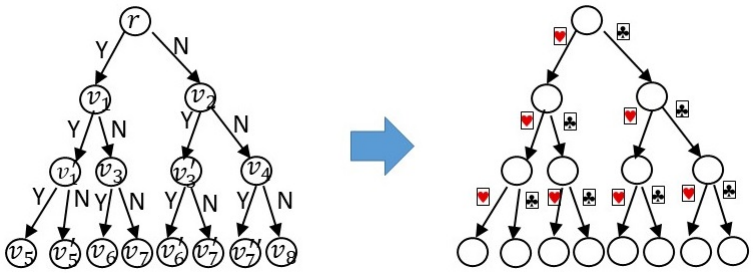

Fig. 4. Setting "yes" and "no" cards.

The randomization is as follows. For each internal node, two subtrees with the "yes" child and the "no" child are randomly swapped.

Protocol 1 *(Randomization of a complete binary decision tree)*

1. *For* $i = 1$ *to* $k - 1$ *do*
2. *For each internal node* v *of depth* i
3. *Alice and Bob randomly swap "yes" and "no" subtree of* v *(Fig. 5)*
4. *Endfor*
5. *Endfor*

In Fig. 5, rectangles are piles of cards. The arrow between the piles means randomly swapping two piles.

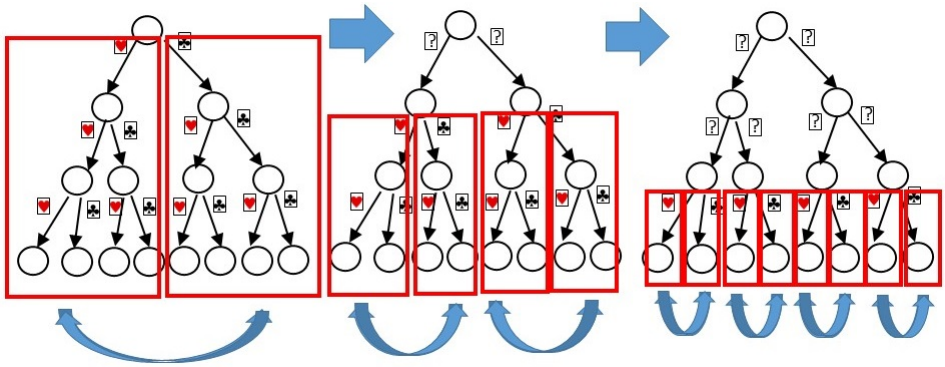

Fig. 5. Randomization of a complete binary tree.

The evaluation by Bob is as follows. First, set v as the root of the tree. Repeat the following step until v becomes a leaf node. Bob privately evaluates $c(v)$ and

opens the two cards on the outgoing edges of v. If the evaluation result is "yes" ("no"), Bob sets v to the child node with $\boxed{\heartsuit}$ ($\boxed{\clubsuit}$) card, respectively. If v is a leaf node, $c(v)$ is the result.

Protocol 2 *(Bob's Evaluation of a complete binary decision tree)*

1. $v = r$ /* *root of the decision tree* */
2. *Repeat*
3. *Bob privately sees cards for two child nodes of* v.
4. *Bob privately sees and evaluates* $c(v)$. *If the answer is "yes" ("no"), Bob sets* v *to the child node with* $\boxed{\heartsuit}$($\boxed{\clubsuit}$), *respectively.*
5. *Until* v *is a leaf.*
6. *Bob privately sees the result* $c(v)$.

Theorem 1. *Protocol 1 and 2 are correct and secure.*

Proof. Correctness: Since the "yes" and "no" edges are known to Bob after shuffles because of $\boxed{\heartsuit}$ and $\boxed{\clubsuit}$ cards, Bob can obtain the correct answer.

Security: Alice obtains no information about Bob's evaluation result at each internal node v_i since the "yes" and "no" edges are randomized. Since the evaluation results are unknown, the final result is also unknown.

The security of Alice's decision tree can be shown as follows. Any trees T_0, T_1 that the adversary initially provides are converted to the same complete binary tree. At the evaluation, the adversary obtains no information other than the traversed path P from the root to the leaf. Since P is the same in T_0 and T_1, the adversary obtains no information about the tree. □

The protocol uses $2^k - 1$ $\boxed{\heartsuit}$ and $\boxed{\clubsuit}$ cards. The total number of shuffles for the randomization is $2^k - 1$. Since the shuffles by the same depth nodes are independent, they can be executed by a single shuffle using some additional tools. Thus the number of shuffles can be reduced to k.

5 Protocols for general decision trees

This section shows the case when copying the condition card $c(v)$ is not easy. The number of internal nodes and leaf nodes might reveal some information about the decision tree. Since the number of internal nodes of a complete binary tree of depth k is $2^k - 1$, if the number of internal nodes is less than $2^k - 1$, Bob knows that there are multiple paths from the root to a leaf node. That might reveal some information about the decision tree. Thus Alice adds dummy nodes to hide the information. The number of leaf nodes gives the number of different final answers. That might be information to be hidden (note that if the final answer is just "yes/no," there is no security problem that Bob knows the number of leaf nodes is two). Thus Alice might want to add dummy leaf nodes to hide the number of final answers.

After adding dummy nodes, the decision tree is changed to $DD = (V = V_1 \cup V_2 \cup V_3 \cup V_4, E)$, where V_3 is the set of dummy internal nodes and V_4 is the

set of dummy leaf nodes. $|V_1|+|V_3| = 2^k - 1$ and $|V_2|+|V_4| = 2^k$. V_3 and V_4 are isolated nodes. Fig. 6 shows an example of adding isolated dummy nodes to the tree in Fig. 2.

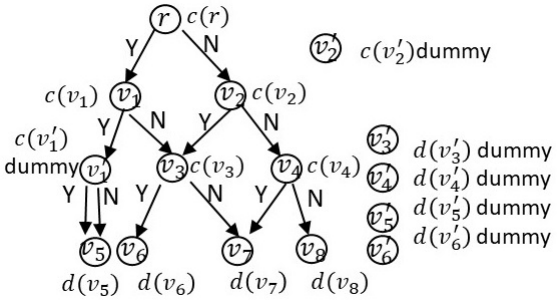

Fig. 6. Adding isolated dummy nodes.

Note that when Bob knows the result is "yes/no", $|V_2| = 2$ and $V_4 = \emptyset$. The protocol for the case can be similarly obtained, thus this section assumes $|V_1|+|V_3| = 2^k-1$ and $|V_2|+|V_4| = 2^k$. Let $n = |V| = 2^{k+1}-1$. The nodes in DD are numbered as 1 to n. Initially, Alice sets the cards to represent the decision tree DD as follows. Alice sets face-down condition card $c(1), c(2), \ldots, c(n)$ in a row. The conditions and the results are written on these cards. Alice sets these cards by the increasing order of the depth. $c(x)(2^i \leq x \leq 2^{i+1}-1)$ are condition cards of the nodes with depth $i(0 \leq i \leq k)$. Below the card $c(x)$, Alice sets two cards $O(2x-1, x)$ and $O(2x, x)$ that represent the "yes" outgoing edge and the "no" outgoing edge of node x. Below the card $c(x)$ of depth $i(0 < i \leq k)$, Alice sets 2^i rows of face-down cards $I(j, x)(2^i - 1 \leq j \leq 2^{i+1} - 2)$. These cards represent incoming edges to node x. Row i consists of one card $O(i, x)$ that represents an outgoing edge from node x and multiple cards $I(i, y)$ that represents an incoming edge to node y. These cards show one directed edge from node x to node y.

Internal node x has two cards that represent the outgoing edges. $O(2x - 1, x) = \boxed{\heartsuit}$, which means "yes" and $O(2x, x) = \boxed{\clubsuit}$, which means "no". If x is a dummy node without a child, $O(2x-1, x)$ and $O(2x, x)$ can be arbitrary, since they are never seen. If "yes" child of node x is node y, $I(2x-1, y) = \boxed{\heartsuit}$. All the other card $I(2x-1, \cdot) = \boxed{\clubsuit}$. If "no" child of node x is node y, $I(2x, y) = \boxed{\heartsuit}$. All the other card $I(2x, \cdot) = \boxed{\clubsuit}$.

If node x is dummy node without a child, $I(2x-1, \cdot)$ and $I(2x, \cdot)$ are arbitrary.

Fig. 7 shows an example of Alice's card set to the decision tree in Fig. 6. The left (right) side below $c(x)$ has $O(\cdot, x)(I(\cdot, x))$, respectively. Note that the cards are set face-up to show the marks in Fig. 7. Alice sets all the cards face-down. $\boxed{\ }$ means the card is arbitrary. In the following figures, the mark of each face-

down card ? is written at the right bottom. If the mark is arbitrary, no mark is written. In Fig. 7, the first row shows "yes" edge (r, v_1). The 6th row shows "no" edge (v_2, v_4). Since dummy node v'_2 has no child, 13th and 14th rows are arbitrary.

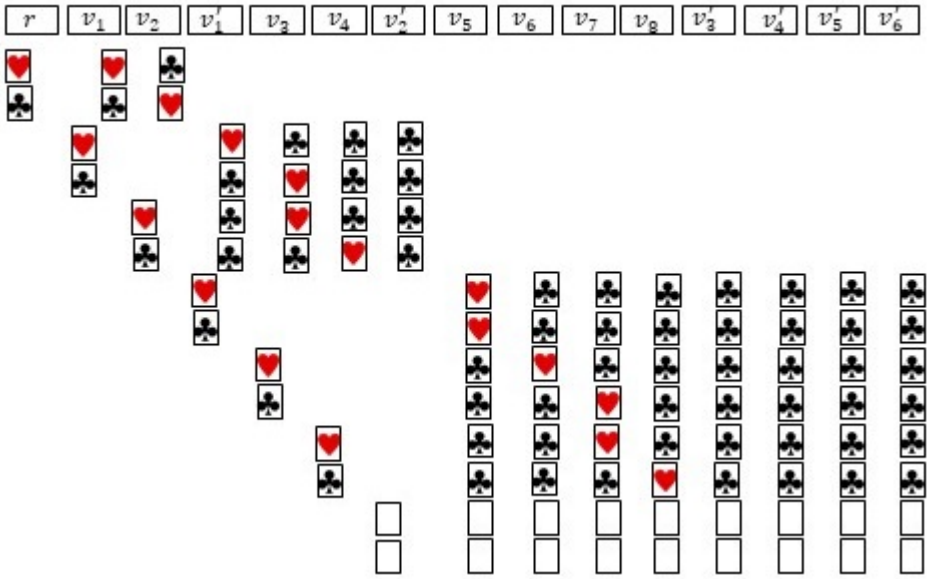

Fig. 7. Cards set by Alice.

First, we show Bob's evaluation steps without security. Bob can evaluate the decision tree using the cards though Alice can know Bob's "yes/no" selection on each node and the final result. Bob sees the condition card of the first node and decides "yes/no". If the result is "yes" ("no"), open the cards $I(1,\cdot)(I(2,\cdot))$, respectively. There is just one ♡ and all other cards are ♣. If $(I(1,j)$ (or $I(2,j))$ $=$ ♡, Bob next sees j-th node. Bob sees the condition card of the j-th node and decides "yes/no". If the result is "yes" ("no"), open the cards $I(2j-1,\cdot)(I(2j,\cdot))$, respectively. Bob repeats the procedure and after the k-th iteration, Bob sees a leaf node. The condition card on the node has the result.

Suppose that $c(r)$ ="no", $c(v_2)$ ="yes", $c(v_3)$ ="no", and the result is v_7. Since $c(r)$="no", Bob sees the second row and opens $I(2,\cdot)$. Since $I(2,3) =$ ♡, Bob knows the next node is the 3rd column. Bob sees $c(v_2)$ and the answer is "yes". The "yes" edge information is in the 5th row, Bob opens $I(5,\cdot)$. Since $I(5,5) =$ ♡, Bob knows the next node is the 5th column. Bob sees $c(v_3)$ and the answer is "no". The "no" edge information is in the 10th row. Bob opens

$I(10,\cdot)$. Since $I(10,10) = \boxed{\heartsuit}$, Bob knows the next node is the 10th column. Bob sees $c(v_7)$ and this is the final result since $k = 3$.

The above procedure outputs the correct result. However, Alice can know Bob's selection of "yes/no" in each node and all the nodes Bob selects. To make the selection by Bob secure, the decision tree must be randomized in advance.

Alice and Bob randomize the cards to hide the information of the nodes Bob accesses to evaluate the decision tree.

Protocol 3 *(Randomization protocol for any binary tree)*

1. */* First phase: row randomization */*
2. *Alice and Bob make pile P_i by the cards in i-th row ($1 \le i \le 2(2^k - 1)$).*
3. *Alice and Bob execute a pile-scramble shuffle on P_{2i-1} and P_{2i} ($1 \le i \le 2^k - 1$).*
4. */* Second phase: internal node randomization in each depth */*
5. *For $i = 1$ to $k - 1$ do*
6. *Alice and Bob make pile P_j by $c(2^i - 1 + j)$ and the cards in $(2^i - 1 + j)$-th column, $(2(2^i - 1 + j) - 1)$-th row, and $(2(2^i - 1 + j))$-th row ($1 \le j \le 2^i$).*
7. *Alice and Bob executes a pile-scramble shuffle on P_j ($1 \le j \le 2^i$).*
8. *Endfor*
9. */* Third phase: leaf node randomization */*
10. *Alice and Bob make pile P_i by $c(2^k - 1 + i)$ and the cards in $(2^k - 1 + i)$-th column ($1 \le i \le 2^k$).*
11. *Alice and Bob executes a pile-scramble shuffle on P_i ($1 \le i \le 2^k$).*

The randomization consists of three phases. The first phase is the randomization of the rows. The odd (even) rows represent "yes" ("no") outgoing edges, respectively. Thus the order of rows must be randomized. After the randomization is executed, Alice cannot know which row is "yes" edge or "no" edge.

Alice and Bob makes $2(2^k-1)$ set of piles $P_i (1 \le i \le 2(2^k-1))$. P_i consists of i-th row. Alice and Bob execute a pile-scramble shuffle on P_{2i-1} and P_{2i}, that is, $(2i-1)$-th row and $2i$-th row are randomly swapped. Fig. 8 shows a row shuffle.

The second phase is the randomization of the sequence of internal nodes. Since Alice knows which node has which condition, the order of internal nodes of the same depth must be randomized while keeping the directed edge information. For each depth $i (1 < i < k)$, execute the following randomization. Alice and Bob make 2^i piles $P_1, P_2, \ldots, P_{2^i}$. $P_j (1 \le j \le 2^i)$ consists of cards for j-th internal node of depth i, that is, node with condition card $c(2^i-1+j)$. Let $x = 2^i-1+j$. The j-th internal node of depth i has cards for incoming edges from depth $(i-1)$ node(s), $I(l,x) (2^{i-1} \le l \le 2^i - 1)$. It has cards for outgoing edges $O(2x-1,x)$, $O(2x,x)$, $I(2x-1,l)$, and $I(2x,l) (2^{i+1} \le l \le 2^{i+2} - 1)$. That is, P_j consists of the cards of x-th column, $(2x-1)$-th row, and $2x$-th row.

Alice and Bob execute a pile-scramble shuffle on $P_j (1 \le j \le 2^i)$. Fig. 9 shows the randomization of depth 1 to the result of swapping $(P_1, P_2), (P_3, P_4), (P_9, P_{10})$ by the randomization of Fig. 8. Fig. 10 shows the randomization of depth 2 to

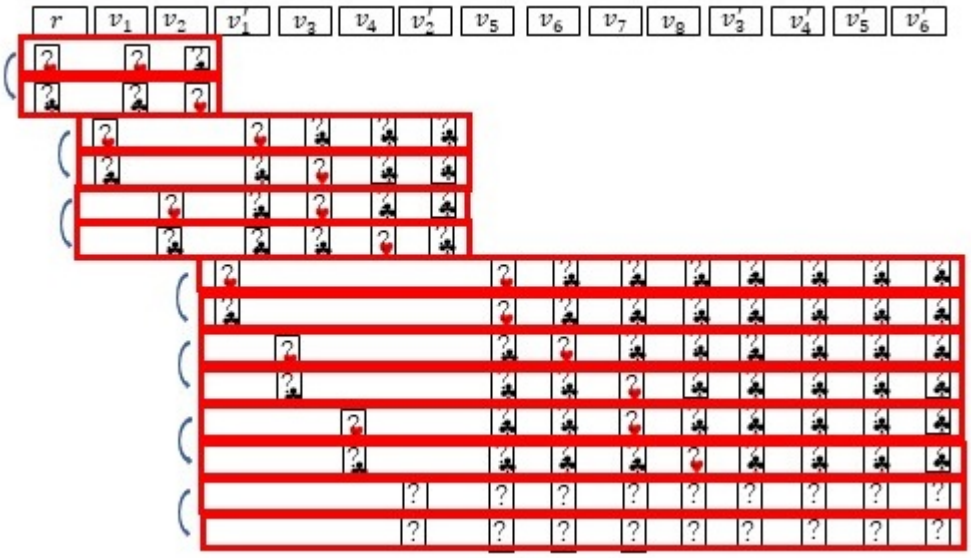

Fig. 8. Row shuffle of the cards.

the result of swapping v_1 and v_2 by the randomization of Fig. 9. In the figures, the polygons shows the piles.

The third phase is the randomization of leaf nodes. The order of leaf nodes must be hidden from Alice. Alice and Bob make 2^k piles $P_1, P_2, \ldots, P_{2^k}$. $P_i (1 \leq i \leq 2^k)$ consists of cards i-th leaf node, that is, node with condition card $c(2^k - 1 + i)$. Let $x = 2^k - 1 + i$. The i-th leaf node has cards for incoming edges from depth $(k-1)$ nodes, $I(l, x)(2^{k-1} \leq l \leq 2^k - 1)$. It has no cards for outgoing edges. That is, P_i consists of the cards of x-th column.

Alice and Bob execute a pile-scramble shuffle on $P_i (1 \leq i \leq 2^k)$. Fig. 11 shows the randomization of leaf nodes to the result with the node sequence v_4, v'_2, v_3, v'_1 by the randomization of Fig. 10.

The evaluation protocol after the randomization is executed as follows.

Protocol 4 *(Bob's evaluation protocol for any binary tree)*

1. *Set* $i = 1$.
2. *For* $s = 1$ *to* k *do*
   Bob privately opens $c(i)$ *and decides "yes/no." Bob privately reveals* $O(\cdot, i)$ *and sees the marks. If the result is "yes" ("no"), Bob sets* j *to be the row that satisfies* $O(j, i) =$ $\boxed{\heartsuit}$($\boxed{\clubsuit}$)*, respectively. Bob publicly opens all the cards* $I(j, \cdot)$*. There is one* $\boxed{\heartsuit}$ *card. Let* i *be the column that satisfies* $I(j, i) =$ $\boxed{\heartsuit}$.
3. *Endfor*
4. */* i must be a leaf node */*
   Bob privately reveals $c(i)$ *and obtains the result.*

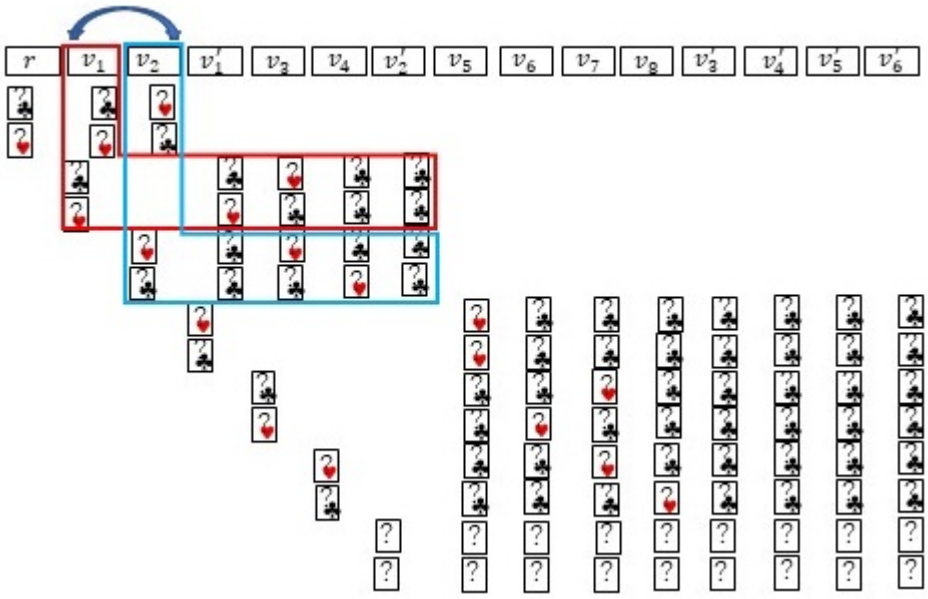

Fig. 9. Randomization of depth 1 nodes.

We show an evaluation example for the randomized table in Fig. 12, in which the leaf node sequence is changed to $v'_3, v'_6, v_7, v_5, v_8, v'_5, v_6, v'_4$ by the randomization of Fig. 11.

Suppose that $c(r)$ ="no", $c(v_2)$ ="yes", $c(v_3)$ ="no", and the result is v_7. Bob first privately reveals $c(r)$ and the evaluation result is "no". Bob privately reveals $O(\cdot, 1)$ and selects the first row since $O(1,1) =$ $\boxed{\clubsuit}$. Bob opens all $I(1,\cdot)$ and sees $I(1,2) =$ $\boxed{\heartsuit}$. Thus the second column is the child node. Bob privately reveals $c(v_2)$ and the evaluation result is "yes". Bob privately reveals $O(\cdot, 2)$ and selects the 3rd row since $O(3,2) =$ $\boxed{\heartsuit}$. Bob opens all cards in $I(3,\cdot)$ and sees $I(3,6) =$ $\boxed{\heartsuit}$. Thus the 6th column is the child node. Bob privately reveals $c(v_3)$ and the evaluation result is "no". Bob privately reveals $O(\cdot, 6)$ and selects the 11th row since $O(11,6) =$ $\boxed{\clubsuit}$. Bob opens all cards in $I(11,\cdot)$ and sees $I(11,10) =$ $\boxed{\heartsuit}$. Thus the 10th column is the child node. Bob privately reveals $c(v_7)$ and obtains the final result.

Note that Alice might have set incorrect cards. In this case, Bob can claim that DD is incorrect with proof. In Step 2, when Bob sees $O(\cdot, i)$, the numbers of $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$ cards might be incorrect. In that case, Bob opens all $O(\cdot, i)$ cards and claims that DD is incorrect. When Bob opens $I(i,\cdot)$, Bob claims that DD is incorrect if the number of $\boxed{\heartsuit}$ cards is not one.

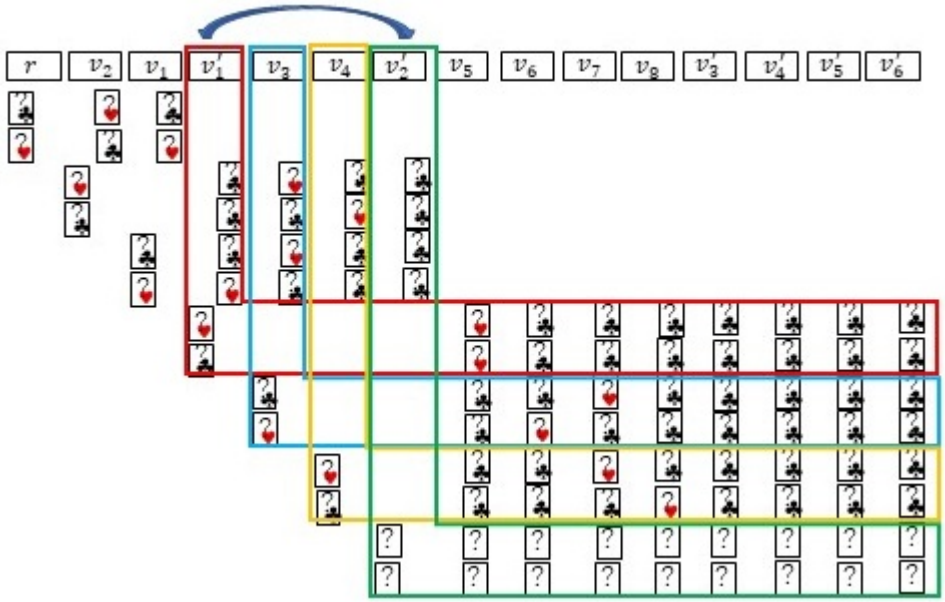

Fig. 10. Randomization of depth 2 nodes.

Since Bob opens all cards of $I(i, \cdot)$, Alice can verify that Bob correctly visits the child node.

When another user Carol wants to use DD, all opened cards are set face-down and they execute the randomizations again before the evaluation.

Theorem 2. *Protocol 3 and 4 are correct and secure.*

Proof. Correctness: Since the "yes" and "no" edges are known to Bob after shuffles because of $\boxed{\heartsuit}$ and $\boxed{\clubsuit}$ cards, Bob can obtain the correct answer.

Security: Alice obtains no information about Bob's evaluation result at each internal node v_i since the "yes" and "no" edges are randomized. Since the evaluation results are unknown, the final result is also unknown.

The security of Alice's decision tree can be shown as follows. Any trees T_0, T_1 that the adversary initially provides are converted to the cards with the same size. At the evaluation, the adversary obtains no information other than the traversed path P from the root to the leaf. Since P is the same in T_0 and T_1, the adversary obtains no information about the tree. □

When the depth is k, the protocol uses $2^{k+1} - 1$ condition cards, $3(2^k - 1)$ $\boxed{\heartsuit}$ cards and $(4^{k+1} - 3 \cdot 2^k - 1)/3$ $\boxed{\clubsuit}$ cards. The total number of shuffles is $2^k + k - 1$. Since the multiple shuffles in the first phase can be parallelly executed, the number of shuffles can be reduced to k.

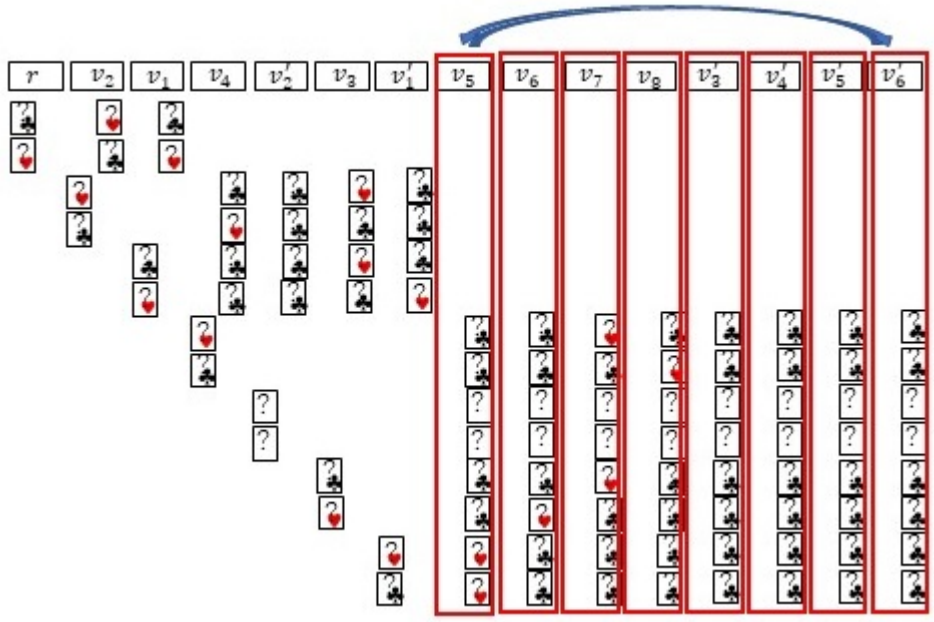

Fig. 11. Randomization of leaf nodes.

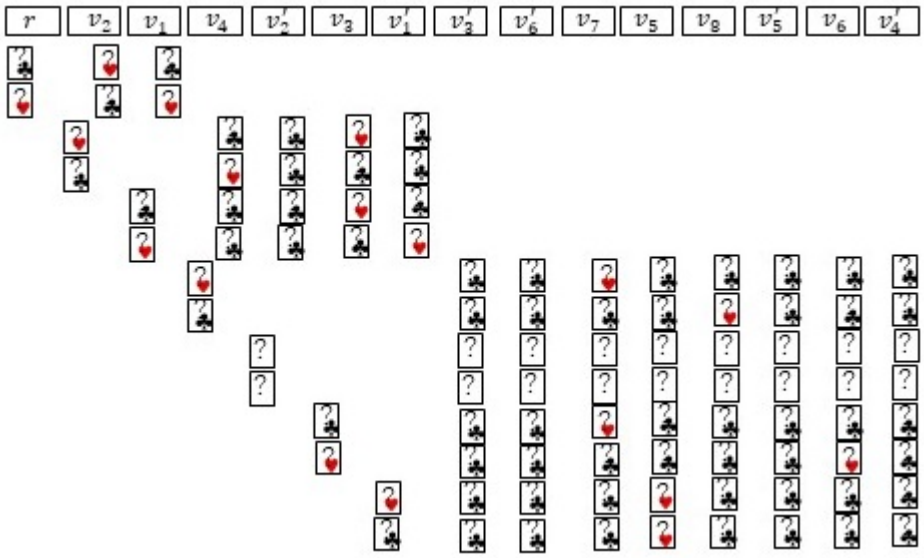

Fig. 12. The result of randomizations.

6 Performance evaluation

This section evaluates the execution times of the proposed protocols and compares them with an existing computer-based protocol. The execution times of primitives in card-based protocols are counted in [7]. Setting one card (called "add" in [7]) and turning one card (called "turn" in [7]) take 0.8 seconds on average. Shuffling cards takes 138.2 seconds on average. Note that the shuffle in [7] is a random bisection cut, which slightly differs from the pile-scramble shuffle, but the execution times are considered to be almost the same.

For the complete binary tree protocol with depth k in Section 4, initially Alice sets all the cards for the nodes and edges. Since the total number of cards ($\boxed{\heartsuit}$, $\boxed{\clubsuit}$, condition cards) is $2^{k+2}-3$, setting needs $0.8\times(2^{k+2}-3)$ sec. Note that at the setting, the cards can be set as the piles of the scrambles, we do not need extra time for the preparation of shuffles. Shuffles are executed k times, thus it takes $138.2k$ seconds. After the shuffles, the cards must be set to the original place again. Thus it takes $0.8\times(2^{k+2}-3)$ seconds. After the setup, Bob privately turns a card, sees the card, and then turns it again for the evaluation. It takes $0.8\times(2(k+1))$ sec. In total, $1.6\times 2^{k+2}+139.8k-3.2$ seconds are necessary. When $k=5$, the time is about 15 minutes. Thus, $k\leq 5$ is the size of reasonable execution time. For the computer-based protocol in [15], the computation time (that excludes key generations and user inputs) is about one minute when the tree is a complete binary tree and $k=5$. The protocol needs extra time for user inputs, thus the total time might be several minutes.

For any tree protocol with depth k in Section 5, a similar evaluation can be done. Setting cards needs $2\times 0.8\times 1/3(4^{k+1}+12\cdot 2^{k+1}-13)$ seconds. Shuffles need $138.2k$ seconds. Evaluation needs $0.8\times(6k+2^{k+1})$ seconds. In total, $1.6\times 1/3(4^{k+1}+15\cdot 2^k+9k-13)+138.2k$ seconds are necessary. When $k=4$, the time is about 20 minutes. Thus, $k\leq 4$ is the size of reasonable execution time. For the computer-based protocol in [15], the computation time (that excludes key generations and user inputs) is about 2 minutes when the tree is not a complete binary tree and $k=4$. The protocol needs extra time for user inputs, thus the total time might be several minutes.

When the tree size is a small constant, the card-based protocols can be executed in a reasonable time.

7 Conclusion

This paper showed a secure evaluation protocol for decision trees. A secure condition evaluation protocol at each internal node is one of the important further studies. Reducing the number of cards is another important problem.

References

1. den Boer, B.: More efficient match-making and satisfiability the five card trick. In: Proc. of EUROCRYPT '89, LNCS Vol. 434. pp. 208–217 (1990)

2. Cheung, E., Hawthorne, C., Lee, P.: Cs 758 project: Secure computation with playing cards (2013), http://cdchawthorne.com/writings/secure_playing_cards.pdf
3. Hanaoka, G., Iwamoto, M., Watanabe, Y., Mizuki, T., Abe, Y., Shinagawa, K., Arai, M., Yanai, N.: Physical and visual cryptography to accelerate social implementation of advanced cryptographic technologies. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences pp. 214–228 (2023), (In Japanese)
4. Ji, K., Zhang, B., Lu, T., Li, L., Ren, K.: Uc secure private branching program and decision tree evaluation. IEEE Transactions on Dependable and Secure Computing (2022)
5. Koch, A., Walzer, S., Härtel, K.: Card-based cryptographic protocols using a minimal number of cards. In: Proc. of Asiacrypt 2015, LNCS Vol. 9452. pp. 783–807 (2015)
6. Manabe, Y.: Survey: Card-based cryptographic protocols to calculate primitives of boolean functions. International Journal of Computer & Software Engineering **27**(1), 178 (2022)
7. Miyahara, D., Ueda, I., Hayashi, Y.i., Mizuki, T., Sone, H.: Evaluating card-based protocols in terms of execution time. International Journal of Information Security **20**(5), 729–740 (2021)
8. Mizuki, T.: Applications of card-based cryptography to education. In: IEICE Technical Report ISEC2016-53. pp. 13–17 (2016), (In Japanese)
9. Mizuki, T., Shizuya, H.: A formalization of card-based cryptographic protocols via abstract machine. International Journal of Information Security **13**(1), 15–23 (2014)
10. Mizuki, T., Shizuya, H.: Computational model of card-based cryptographic protocols and its applications. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **100**(1), 3–11 (2017)
11. Mizuki, T., Sone, H.: Six-card secure and and four-card secure xor. In: Proc. of 3rd International Workshop on Frontiers in Algorithms(FAW 2009), LNCS Vol. 5598. pp. 358–369 (2009)
12. Nakai, T., Misawa, Y., Tokushige, Y., Iwamoto, M., Ohta, K.: How to solve millionaires' problem with two kinds of cards. New Generation Computing **39**(1), 73–96 (2021)
13. Ono, H., Manabe, Y.: Efficient card-based cryptographic protocols for the millionaires' problem using private input operations. In: Proc. of 13th Asia Joint Conference on Information Security(AsiaJCIS 2018). pp. 23–28 (2018)
14. Ono, H., Manabe, Y.: Card-based cryptographic logical computations using private operations. New Generation Computing **39**(1), 19–40 (2021)
15. Saito, Y., Ogata, W.: Private decision tree evaluation by a single untrusted server for machine learning as a service. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **105**(3), 203–213 (2022)
16. Today, M.: Clinical flowcharts (2023), https://medicinetoday.com.au/clinical-flowcharts [Accessed: (Sep 05, 2024)]
17. Tsuchida, H., Nishide, T., Maeda, Y.: Private decision tree evaluation with constant rounds via (only) ss-3pc over ring. In: Provable and Practical Security: 14th International Conference, ProvSec 2020, Singapore, November 29–December 1, 2020, Proceedings 14. pp. 298–317. Springer (2020)