AUSTRALASIAN JOURNAL OF COMBINATORICS
Volume **36** (2006), Pages 279–293

Securely computing XOR with 10 cards

Takaaki Mizuki   Fumishige Uchiike   Hideaki Sone

Information Synergy Center, Tohoku University
Aramaki-Aza-Aoba 6–3, Aoba-ku
Sendai 980–8578
Japan
tm-paper@rd.isc.tohoku.ac.jp

Abstract

There exist several protocols by which Alice and Bob can securely compute some function using a deck of cards. The most efficient protocol currently known is given by Stiglic; his elaborate but simple protocol can allow Alice and Bob to securely compute the AND function using only 8 cards. The existence of the AND protocol implies that one can design a protocol which securely computes any Boolean function; for example, the XOR function can be securely computed using 12 cards, although the computation requires a relatively large task such as copying the input and repeating the AND protocol. This paper addresses efficient secure computation of XOR, and gives a simple protocol which securely computes the XOR function using only 10 cards.

1 Introduction

Assume that Alice and Bob, who are honest-but-curious, have secret bits $a \in \{0,1\}$ and $b \in \{0,1\}$, respectively. They wish to *securely compute* some Boolean function $f(a,b)$ such as AND, OR and XOR: they wish to learn the value of $f(a,b)$, but do not want to reveal their own secret bits more than necessary. It has been known for many years that such secure computations can be implemented by a *deck of cards*, and there exist several protocols to achieve it. In this paper, we give another new protocol for some secure computation. Briefly summarizing our results, we give a protocol by which Alice and Bob can securely compute the XOR function $a \oplus b$ using a deck of 10 cards; our protocol produces its output in a "committed format," i.e. the output is obtained in a hidden way so that it can be used later for another computation.

This paper begins with an overview of the classic protocol, called the "five-card trick [3]," which securely computes the AND function $a \wedge b$.

1.1 The "five-card trick"

Assume that Alice and Bob wish to securely compute the conjunction $a \wedge b$ of their secret bits. Note that, after succeeding in such secure computation and learning the value of $a \wedge b$, Alice or Bob may know the other's bit depending on the value of a or b (but we do not care); for example, since $1 \wedge b = b$, Alice gets to know Bob's bit b whenever $a = 1$; whereas, if $a = 0$, then Alice never learns b. To achieve such secure computation of AND, Boer [3] proposed an elegant protocol, which uses a deck of five cards: three $\boxed{\clubsuit}$'s and two $\boxed{\heartsuit}$'s. Before going into the detail of the protocol, we first mention the properties of cards appearing in this paper.

All cards of the same type ($\boxed{\clubsuit}$ or $\boxed{\heartsuit}$) are assumed to be indistinguishable from one another. We denote by $\boxed{?}$ a card with its face down. We also assume that the back side $\boxed{?}$ of each card is identical, as usual. To deal with Boolean values, we use the following encoding (throughout this paper):

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

We now explain the protocol. First, according to Alice's bit a and the negation $\bar{b}$ of Bob's bit b, they put the five cards as follows:

$$\underbrace{\boxed{?}\boxed{?}}_{a}\boxed{\clubsuit}\underbrace{\boxed{?}\boxed{?}}_{\bar{b}}.$$

That is, one of the three $\boxed{\clubsuit}$'s is centered, Alice puts two cards of different types face down at the left so that the order (which Bob must not see) is

$$\begin{cases} \boxed{\clubsuit}\boxed{\heartsuit} & \text{if } a = 0 \\ \boxed{\heartsuit}\boxed{\clubsuit} & \text{if } a = 1, \end{cases}$$

and Bob puts the remaining two cards face down at the right so that the order (which Alice must not see) is

$$\begin{cases} \boxed{\heartsuit}\boxed{\clubsuit} & \text{if } b = 0 \text{ (i.e. } \bar{b} = 1) \\ \boxed{\clubsuit}\boxed{\heartsuit} & \text{if } b = 1 \text{ (i.e. } \bar{b} = 0). \end{cases}$$

We call such a sequence of two face-down cards a *commitment*; the sequence of the two leftmost cards is a commitment to a (by Alice), and the sequence of the two rightmost cards is a commitment to $\bar{b}$ (by Bob). Then, Alice and Bob turn the centered card $\boxed{\clubsuit}$ face down, and apply a *random cut*, which is denoted by $\langle\cdot\rangle$:

$$\underbrace{\boxed{?}\boxed{?}}_{a}\boxed{?}\underbrace{\boxed{?}\boxed{?}}_{\bar{b}} \rightarrow \Big\langle\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\Big\rangle \rightarrow \boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}.$$

A random cut (which is also called a *random cyclic shuffling*) means that, as in the case of usual card games, a random number of leftmost cards is moved to the right

	committed format?	# of types	# of cards	avg. # of trials	need copying?
∘ *Secure computation of AND*					
Boer [3] (§1.1)	no	2	5	—	no
Crépeau, Kilian [4]	yes	4	10	6 (3)	no
Niemi, Renvall [10]	yes	2	12	2.5	yes
Stiglic [13] (§2.1)	yes	2	8	2	no
∘ *Secure computation of XOR*					
Crépeau, Kilian [4]	yes	4	14	6 (3)	yes
using Stiglic's (§3)	yes	2	12	6	yes
ours (§4)	yes	2	10	2	no

Table 1: The known protocols and ours with their performance.

without changing their order (so that, of course, the random number must not be known to Alice or Bob); to implement this, it suffices that Alice and Bob take turns at cutting the deck until they are satisfied. Finally, Alice and Bob open all the five cards. Then, the resulting sequence is either

$$\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{\heartsuit} \text{ or } \boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit} \tag{2}$$

apart from cyclic rotations, i.e. either the two $\boxed{\heartsuit}$'s are "cyclically" next to each other or not. One can easily verify that the former case implies $a \wedge b = 1$, and the latter case implies $a \wedge b = 0$.

Thus, the "five-card trick" can securely compute $a \wedge b$ quite efficiently. Its only drawback is that the format of its output $a \wedge b$ differs from the format of input a and b (remember the encoding rule (1) and the resulting sequences (2)). If one can have a protocol such that its output $a \wedge b$ is obtained in a *committed format*, i.e. its output is described as a sequence

$$\underbrace{\boxed{?}\boxed{?}}_{a\wedge b}$$

which follows the encoding rule (1) (and Alice and Bob have no knowledge about the value than their own secret bits), then the output can be later used as input for another computation. Hence, such a protocol is much desirable. Indeed, since the invention of the "five-card trick," some protocols which produce their output in a committed format have been developed.

1.2 The history

Table 1 indicates the known protocols and their performance.

As explained in Section 1.1, Boer [3] gave a secure AND protocol, called the "five-card trick," which uses three $\boxed{\clubsuit}$'s and two $\boxed{\heartsuit}$'s.

To overcome the drawback of the "five-card trick," Crépeau and Kilian [4] developed a secure AND protocol, which produces its output in a committed format using three $\boxed{\clubsuit}$'s, three $\boxed{\heartsuit}$'s, two $\boxed{\spadesuit}$'s and two $\boxed{\diamondsuit}$'s. Thus, the protocol requires four types of cards, and uses 10 cards in total. Furthermore, it is a *Las Vegas algorithm*, and takes an average of six trials. (We note that the average number of trials can be decreased to three by slightly improving the output positions in the protocol.)

Crépeau and Kilian also gave a secure XOR protocol by modifying their AND protocol. Their XOR protocol, unlike their AND protocol, needs to *copy* a commitment to Bob's bit b (a method for copying was also given by them [4], and will be presented in Section 2.2.3), and consequently requires four additional cards: two $\boxed{\clubsuit}$'s and two $\boxed{\heartsuit}$'s (provided that "reusable" cards are utilized). Thus, Crépeau-Kilian XOR protocol requires 14 cards in total.

Next, Niemi and Renvall [10] constructed a secure AND protocol by making use of the sequence in the "five-card trick" twice. Their AND protocol needs only two types of cards, and requires 12 cards (six $\boxed{\clubsuit}$'s and six $\boxed{\heartsuit}$'s) in total. Furthermore, it takes an average of 2.5 trials.

Finally, Stiglic [13] proposed an efficient secure AND protocol (the detail of which will be explained in Section 2.1). His elaborate but simple protocol requires only 8 cards (four $\boxed{\clubsuit}$'s and four $\boxed{\heartsuit}$'s), and needs no copy of input. Furthermore, it takes an average of two trials. Thus, Stiglic's AND protocol is the most efficient one currently known.

1.3 Our results and related work

This paper addresses efficient secure computation of the XOR function $a \oplus b$, and we will give a new secure XOR protocol, which uses only 10 cards: five $\boxed{\clubsuit}$'s and five $\boxed{\heartsuit}$'s. Our XOR protocol needs no copy of input, and takes an average of two trials. Thus, Our XOR protocol is more efficient than Crépeau-Kilian XOR protocol, as designated in Table 1.

Since securely computing the NOT operator is trivial (as will be seen in Section 2.2.1), the existence of a secure AND protocol producing its output in a committed format immediately implies that one can easily design a protocol which securely computes any Boolean function, such as the OR function $a \vee b = \overline{\bar{a} \wedge \bar{b}}$. Therefore, the XOR function can be also securely computed according to the identity $a \oplus b = (a \wedge \bar{b}) \vee (\bar{a} \wedge b)$. However, even if we adopt Stiglic's AND protocol (which is the most efficient one currently known), such secure computation of XOR needs a relatively large task: it needs to copy commitments to the input a and b, needs to repeat the AND protocol three times, requires 12 cards in total, and takes an average of six trials, as will be shown in Section 3. Thus, Our XOR protocol is more efficient than the protocol constructed in such a way (see Table 1).

All the protocols appearing in Table 1 work without any use of computers; all they need is just a (small) deck of cards. Thus, our work falls into the area of *designing cryptographic protocols without computers.* As another physical cryptographic tool, Balogh, Csirik, Ishai and Kushilevitz [2] considered a "PEZ dispenser," which securely computes any function without randomization. Recently, Moran and Naor [9] proved that "scratch-off cards" can be used to construct a protocol for bit commitment and coin flipping, but not oblivious transfer. Niemi and Renvall [10], and Salomaa [11, 12] pointed out various situations in which cryptographic protocols without computers may be more effective than those with computers such as ordinary public-key cryptosystems. It should be noted that, as Salomaa mentioned in [11], the area of designing cryptographic protocols without computers has not been investigated so much. Somewhat related to our work is the problem of sharing a secret key by using a random deal of cards [6, 7, 8], or "Russian Cards" problem [1, 5]; however, in the problems, a deck of cards is used as a mathematical model of initial information given to each player, rather than a physical cryptographic tool.

The remainder of the paper is organized as follows. In Section 2, we present some of the known protocols with our interpretations. In Section 3, we mention that this paper addresses secure computation of XOR. In Section 4, we give a simple efficient secure XOR protocol. This paper concludes in Section 5 with some discussions and open problems.

2 Known Protocols with Our Interpretations

In this section, we present some of the known protocols with our interpretations (which lead to some part of the idea behind the construction of our efficient secure XOR protocol).

In Section 2.1, we present Stiglic's AND protocol, which is the most efficient protocol currently known. In Section 2.2, we present other known useful primitives: securely computing NOT, securely computing OR, and copying a commitment. The descriptions of these known protocols are slightly modified in this paper for the readability of the succeeding sections; of course, their substance is not changed.

2.1 Stiglic's AND protocol

Stiglic's AND protocol [13] can allow Alice and Bob to securely compute the conjunction $a \wedge b$ of their secret bits using only 8 cards: four $\boxed{\clubsuit}$'s and four $\boxed{\heartsuit}$'s; furthermore, its output is obtained in a committed format.

1. Place the 8 cards as below, and then turn over all the face-up cards:

$$\boxed{\clubsuit}\underbrace{\boxed{?}\boxed{?}}_{a}\boxed{\clubsuit}\boxed{\heartsuit}\underbrace{\boxed{?}\boxed{?}}_{b}\boxed{\heartsuit} \rightarrow \boxed{?}\underbrace{\boxed{?}\boxed{?}}_{a}\boxed{?}\boxed{?}\underbrace{\boxed{?}\boxed{?}}_{b}\boxed{?}.$$

2. Apply a random cut:

$$\left\langle\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\right\rangle \rightarrow \boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}.$$

3. Open the first and second cards (namely, the two leftmost cards).
   (a) If the resulting sequence is $\boxed{\clubsuit}\boxed{\clubsuit}$ or $\boxed{\heartsuit}\boxed{\heartsuit}$, then go to step 4.
   (b) If the resulting sequence is $\boxed{\clubsuit}\boxed{\heartsuit}$ or $\boxed{\heartsuit}\boxed{\clubsuit}$, then open the third (leftmost) card.
      i. If the sequence of the three face-up cards is $\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}$ or $\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}$, then go to step 4.
      ii. If the sequence of the three face-up cards is $\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}$ or $\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}$, then turn back over the three cards and return to step 2.

4. Depending on the (two or three) face-up cards, the output $a \wedge b$ is obtained as follows:

$$\boxed{\clubsuit}\boxed{\clubsuit}\boxed{?}\underbrace{\boxed{?}\boxed{?}}_{a\wedge b}\boxed{?}\boxed{?}\boxed{?},$$

$$\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{?}\underbrace{\boxed{?}\boxed{?}}_{a\wedge b}\boxed{?}\boxed{?},$$

$$\boxed{\heartsuit}\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\underbrace{\boxed{?}\boxed{?}}_{a\wedge b}\boxed{?}, \text{ or}$$

$$\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{?}\boxed{?}\boxed{?}\underbrace{\boxed{?}\boxed{?}}_{a\wedge b}.$$

Note that steps 2 and 3 in the protocol are repeated until a sequence $\boxed{\clubsuit}\boxed{\clubsuit}$ or $\boxed{\heartsuit}\boxed{\heartsuit}$ is found. Therefore, one can observe that Stiglic's AND protocol takes an average of two trials. (Thus, it is a Las Vegas algorithm.)

As mentioned above, the protocol searches for a sequence $\boxed{\clubsuit}\boxed{\clubsuit}$ or $\boxed{\heartsuit}\boxed{\heartsuit}$, which tells the position of the output $a \wedge b$. For example, consider the case where $\boxed{\clubsuit}\boxed{\clubsuit}$ is found (the case for $\boxed{\heartsuit}\boxed{\heartsuit}$ is similar). Then, the original sequence of the 8 cards was either

$$\overset{*}{\boxed{\clubsuit}}\underbrace{\overset{*}{\boxed{\clubsuit}}\boxed{\heartsuit}}_{a=0}\underbrace{\overbrace{\boxed{\clubsuit}\boxed{\heartsuit}}^{\text{output}}}_{0}\underbrace{\boxed{?}\boxed{?}}_{b}\boxed{\heartsuit} \quad\text{or}\quad \boxed{\clubsuit}\underbrace{\boxed{\heartsuit}\overset{*}{\boxed{\clubsuit}}}_{a=1}\overset{*}{\boxed{\clubsuit}}\boxed{\heartsuit}\underbrace{\overbrace{\boxed{?}\boxed{?}}^{\text{output}}}_{b}\boxed{\heartsuit}.$$

In the former case, we have $a = 0$ and hence a commitment to the output $0 \wedge b = 0$ is the two cards gained by skipping one card from $\boxed{\clubsuit}\boxed{\clubsuit}$. Similarly, in the latter case, we have $a = 1$ and hence the position of the output $1 \wedge b = b$ is acquired correctly.

It should be noted that, after a commitment to the output $a \wedge b$ is obtained, the remaining six cards are *reusable* for executing another computation (provided that the face-down cards are turned over after some shuffling).

2.2 Other primitives

In addition to Stiglic's AND protocol, we now present the known useful primitives: securely computing NOT, securely computing OR, and copying a commitment.

2.2.1 Securely computing NOT

The secure computation of the NOT operator is trivial; given a commitment, reverse the order of the two cards:

$$\underbrace{\boxed{?}\boxed{?}}_{x} \rightarrow \overset{\rightleftharpoons}{\overbrace{\boxed{?}\boxed{?}}} \rightarrow \underbrace{\boxed{?}\boxed{?}}_{\bar{x}}.$$

2.2.2 Securely computing OR

The OR function $a \vee b$ can be securely computed by applying both Stiglic's AND protocol and the NOT protocol above according to "de Morgan's laws" $a \vee b = \overline{\bar{a} \wedge \bar{b}}$ [13].

Alternatively, we note that one can also have a secure OR protocol by slightly modifying the output positions in step 4 of Stiglic's AND protocol as follows:

$$\boxed{\clubsuit}\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\underbrace{\boxed{?}\boxed{?}}_{a\vee b}\boxed{?}, \qquad \boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\boxed{?}\boxed{?}\boxed{?}\underbrace{\boxed{?}\boxed{?}}_{a\vee b},$$

$$\boxed{\heartsuit}\boxed{\heartsuit}\boxed{?}\underbrace{\boxed{?}\boxed{?}}_{a\vee b}\boxed{?}\boxed{?}\boxed{?}, \text{ or } \boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{?}\underbrace{\boxed{?}\boxed{?}}_{a\vee b}\boxed{?}\boxed{?}.$$

Thus, we point out that Stiglic's AND protocol brings not only a commitment to $a \wedge b$ but also a commitment to $a \vee b$.

2.2.3 Copying a commitment

The following protocol [4, 10] makes k copies of a given commitment to $x \in \{0, 1\}$.

1. Put an alternation of $2k+4$ $\boxed{\clubsuit}$'s and $\boxed{\heartsuit}$'s to the right of the given commitment:

$$\underbrace{\boxed{?}\boxed{?}}_{x}\overbrace{\underbrace{\boxed{\clubsuit}\boxed{\heartsuit}}_{0}\underbrace{\boxed{\clubsuit}\boxed{\heartsuit}}_{0}\underbrace{\boxed{\clubsuit}\boxed{\heartsuit}}_{0}\cdots\underbrace{\boxed{\clubsuit}\boxed{\heartsuit}}_{0}}^{2k+4\text{ cards}}.$$

2. Turn over the $2k+4$ face-up cards, and apply a random cut to them:

$$\underbrace{\boxed{?}\boxed{?}}_{x} \quad \left\langle\underbrace{\boxed{?}\boxed{?}}_{0}\underbrace{\boxed{?}\boxed{?}}_{0}\underbrace{\boxed{?}\boxed{?}}_{0}\cdots\underbrace{\boxed{?}\boxed{?}}_{0}\right\rangle.$$

Then, we have

$$\underbrace{\boxed{?}\boxed{?}}_{x}\underbrace{\boxed{?}\boxed{?}}_{r}\underbrace{\boxed{?}\boxed{?}}_{r}\underbrace{\boxed{?}\boxed{?}}_{r}\cdots\underbrace{\boxed{?}\boxed{?}}_{r},$$

where r is a random bit because of the random cut.

3. Apply a random cut to the four leftmost cards:

$$\left\langle \underbrace{\boxed{?}\boxed{?}}_{x}\underbrace{\boxed{?}\boxed{?}}_{r} \right\rangle \;\; \underbrace{\boxed{?}\boxed{?}}_{r}\underbrace{\boxed{?}\boxed{?}}_{r}\cdots\underbrace{\boxed{?}\boxed{?}}_{r},$$

   and then open the first and third cards:

$$\overbrace{\boxed{?}}^{\text{open}}\boxed{?}\overbrace{\boxed{?}}^{\text{open}}\boxed{?}\;\; \overbrace{\underbrace{\boxed{?}\boxed{?}}_{r}\underbrace{\boxed{?}\boxed{?}}_{r}\cdots\underbrace{\boxed{?}\boxed{?}}_{r}}^{2k+2\text{ cards}}.$$

4. If the resulting sequence of the two face-up cards is $\boxed{\clubsuit}\boxed{\clubsuit}$ or $\boxed{\heartsuit}\boxed{\heartsuit}$, then it implies $x = r$, and hence the $2k+2$ rightmost cards immediately form a commitment to x and its k copies:

$$\overbrace{\underbrace{\boxed{?}\boxed{?}}_{x\ (=r)}\underbrace{\boxed{?}\boxed{?}}_{x\ (=r)}\cdots\underbrace{\boxed{?}\boxed{?}}_{x\ (=r)}}^{2k+2\text{ cards}}.$$

   If the resulting sequence of the two face-up cards is $\boxed{\clubsuit}\boxed{\heartsuit}$ or $\boxed{\heartsuit}\boxed{\clubsuit}$, then it implies $x \neq r$, i.e. $x = \bar{r}$, and hence reversing the order of each pair of the two cards brings a commitment to x and its k copies as follows:

$$\overbrace{\underbrace{\boxed{?}\boxed{?}}_{r}}^{\rightleftharpoons}\overbrace{\underbrace{\boxed{?}\boxed{?}}_{r}}^{\rightleftharpoons}\cdots\overbrace{\underbrace{\boxed{?}\boxed{?}}_{r}}^{\rightleftharpoons} \;\rightarrow\; \overbrace{\underbrace{\boxed{?}\boxed{?}}_{x\ (=\bar{r})}\underbrace{\boxed{?}\boxed{?}}_{x\ (=\bar{r})}\cdots\underbrace{\boxed{?}\boxed{?}}_{x\ (=\bar{r})}}^{2k+2\text{ cards}}.$$

Note that, after k copies of the given commitment are obtained, the remaining four (leftmost) cards are reusable for executing another computation.

3 Securely Computing XOR

We are now ready to mention our problem and results. As stated before, this paper deals with secure computation of the XOR function $a \oplus b$.

Note that, once Alice and Bob learn the value of the exclusive-or $a \oplus b$ of their secret bits, each of them gets to know the other's bit completely, because $b = (a \oplus b) \oplus a$ and $a = (a \oplus b) \oplus b$. Therefore, one may feel, at first glance, that secure computation of XOR makes no sense. However, it is not the case. When one has a secure XOR protocol producing its output $a \oplus b$ in a committed format, it can be used as a basic gate for composing larger circuits. For example, using such a protocol, secure multiparty computation of $x_1 \oplus x_2 \oplus \cdots \oplus x_n$ can be done, i.e. n players can learn the parity of their secret bits without leaking any further information. Thus, designing a secure (2-variable) XOR protocol is much important. For another example, consider the case where Alice and Bob want to slip the result $a \oplus b$ to some third party, then even just secure computation of 2-variable XOR is quite useful.

In this paper, we will give a simple and efficient secure XOR protocol which produces its output in a committed format using only 10 cards, as will be seen in the succeeding section.

Since $a \oplus b = (a \wedge \bar{b}) \vee (\bar{a} \wedge b)$, one can construct a secure XOR protocol by using Stiglic's AND protocol and the NOT protocol. However, such a protocol requires a relatively large task: it needs to copy commitments to the input a and b, and needs to repeat the AND protocol three times, due to the formula $(a \wedge \bar{b}) \vee (\bar{a} \wedge b)$. We now count how many cards are necessary for such computation. First, in addition to the four cards necessary for the input a and b, copying the commitment to a requires six additional cards. After a copy of a is obtained, we have four reusable cards (as mentioned in Section 2.2.3). Therefore, when we next copy the commitment to b, only two additional cards are required. Hence, in total, such a XOR protocol requires 12 cards. Furthermore, it takes an average of six trials, because Stiglic's AND protocol must be repeated three times. Thus, Our XOR protocol is more efficient than the protocol constructed according to $a \oplus b = (a \wedge \bar{b}) \vee (\bar{a} \wedge b)$ (see Table 1 again).

4 Our Secure XOR Protocol

In this section, we give our protocol by which Alice and Bob can securely compute the exclusive-or $a \oplus b$ of their secret bits using only 10 cards.

We first outline our protocol in Section 4.1. We next give its building blocks in Section 4.2. We finally describe our protocol in Section 4.3.

4.1 Outline

The main idea behind our XOR protocol is based on the identity

$$a \oplus b = a \oplus b \oplus r \oplus r,$$

where r is a random bit. In order to compute $a \oplus b$, we first compute $a \oplus b \oplus r$ in a *non-committed format*, which means that, given three commitments to three unknown bits a, b and r, we make only the value of $a \oplus b \oplus r$ public. Then, according to the result, we output

$$\begin{cases} r & \text{if } a \oplus b \oplus r = 0 \\ \bar{r} & \text{if } a \oplus b \oplus r = 1 \end{cases}$$

in a committed format. In other words, we reduce our goal (which is securely computing $a \oplus b$ in a committed format) to securely computing $a \oplus b \oplus r$ in a non-committed format. A more detailed outline is below.

1. Generate two commitments to a random bit r which Alice and Bob must not see:

$$\underbrace{\boxed{\clubsuit}\boxed{\heartsuit}}_{0}\underbrace{\boxed{\clubsuit}\boxed{\heartsuit}}_{0} \to \left\langle \boxed{?}\boxed{?}\boxed{?}\boxed{?} \right\rangle \to \underbrace{\boxed{?}\boxed{?}}_{r}\underbrace{\boxed{?}\boxed{?}}_{r}.$$

   (Remember step 2 of the copy protocol described in Section 2.2.3.) The first commitment to the random bit r will be used to compute $a \oplus b \oplus r$ in step 2, and the second one will be used to output either r or $\bar{r}$ in step 3.

2. Using the three commitments

$$\underbrace{\boxed{?}\boxed{?}}_{a}\ \underbrace{\boxed{?}\boxed{?}}_{b}\ \underbrace{\boxed{?}\boxed{?}}_{r}$$

   (and two additional cards), securely compute $a \oplus b \oplus r$ in a non-committed format. This can be done by our building block which will be given in Section 4.2.2.

3. According to the resulting value of $a \oplus b \oplus r$ above, make a commitment to the output $a \oplus b$ by using the second commitment to r, as follows. If $a \oplus b \oplus r = 0$, then the commitment to r immediately forms the output:

$$\underbrace{\boxed{?}\boxed{?}}_{a\oplus b\ (=r)}.$$

   If $a \oplus b \oplus r = 1$, then reversing the two cards of the commitment to r brings the output:

$$\underbrace{\boxed{?}\boxed{?}}_{r} \to \overbrace{\boxed{?}\boxed{?}}^{\rightleftharpoons} \to \underbrace{\boxed{?}\boxed{?}}_{a\oplus b\ (=\bar{r})}.$$

Note that, even if one learns the value of $a \oplus b \oplus r$ in step 2, one never learns anything about each value of a and b, because r is a random secret bit. Thus, by the observation above, securely computing $a \oplus b \oplus r$ in a non-committed format suffices for our goal (which is securely computing $a \oplus b$ in a committed format).

4.2 Building blocks

As mentioned above, we wish to securely compute $a \oplus b \oplus r$, i.e. the 3-variable XOR function, in a non-committed format. To this end, we first observe how to securely compute the (2-variable) XOR function $x_1 \oplus x_2$ in a non-committed format in Section 4.2.1. Then, using the idea, we give a protocol which securely computes the 4-variable XOR function $x_1 \oplus x_2 \oplus x_3 \oplus x_4$ in a non-committed format in Section 4.2.2. Note that such a protocol immediately achieves the end, because $a \oplus b \oplus r$ is equal to $a \oplus b \oplus 0 \oplus r$ (which can be securely computed in a non-committed format by such a 4-variable XOR protocol).

4.2.1 Non-committed computation of 2-variable XOR

Given commitments to x_1 and x_2, apply a random cut, and then open the first and third cards:

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\underbrace{\boxed{?}\boxed{?}}_{x_1} \rightarrow \left\langle \boxed{?}\boxed{?}\boxed{?}\boxed{?} \right\rangle \rightarrow \overbrace{\boxed{?}}^{\text{open}}\boxed{?}\overbrace{\boxed{?}}^{\text{open}}\boxed{?}.$$

If the resulting sequence is $\boxed{\clubsuit}\boxed{\clubsuit}$ or $\boxed{\heartsuit}\boxed{\heartsuit}$, then it implies $x_1 \oplus x_2 = 0$. If the resulting sequence is $\boxed{\clubsuit}\boxed{\heartsuit}$ or $\boxed{\heartsuit}\boxed{\clubsuit}$, then it implies $x_1 \oplus x_2 = 1$. (Remember steps 3 and 4 of the copy protocol described in Section 2.2.3.)

4.2.2 Non-committed computation of 4-variable XOR

Given commitments to x_1, x_2, x_3 and x_4, the following protocol securely computes $x_1 \oplus x_2 \oplus x_3 \oplus x_4$ in a non-committed format.

1. Put the four commitments as follows:

$$\underbrace{\boxed{?}\boxed{?}}_{x_1}\underbrace{\boxed{?}\boxed{?}}_{x_2}\underbrace{\boxed{?}\boxed{?}}_{x_3}\underbrace{\boxed{?}\boxed{?}}_{x_4}.$$

2. Apply a random cut to each of both the four leftmost cards and the four rightmost cards:

$$\begin{array}{cc} \left\langle \boxed{?}\boxed{?}\boxed{?}\boxed{?} \right\rangle & \left\langle \boxed{?}\boxed{?}\boxed{?}\boxed{?} \right\rangle \\ \downarrow & \downarrow \\ \boxed{?}\boxed{?}\boxed{?}\boxed{?} & \boxed{?}\boxed{?}\boxed{?}\boxed{?}. \end{array}$$

3. Draw the first and fifth cards without opening them:

$$\overbrace{\boxed{}}^{\text{drawn}}\boxed{?}\boxed{?}\boxed{?} \quad \overbrace{\boxed{}}^{\text{drawn}}\boxed{?}\boxed{?}\boxed{?}, \tag{3}$$

apply a random cut to the two drawn cards:

$$\left\langle \boxed{?}\boxed{?} \right\rangle \rightarrow \boxed{?}\boxed{?},$$

and then open the two cards.

   (a) If the resulting sequence is $\boxed{\clubsuit}\boxed{\heartsuit}$ or $\boxed{\heartsuit}\boxed{\clubsuit}$, then go to step 4.

   (b) If the resulting sequence is $\boxed{\clubsuit}\boxed{\clubsuit}$ or $\boxed{\heartsuit}\boxed{\heartsuit}$, then return the two drawn cards to the sequence (3) with their faces down, and go back to step 2.

4. Open the third and seventh cards:

$$\boxed{}\boxed{?}\overbrace{\boxed{?}}^{\text{open}}\boxed{?}\quad\boxed{}\boxed{?}\overbrace{\boxed{?}}^{\text{open}}\boxed{?}.$$

If the resulting sequence is $\boxed{\clubsuit}\boxed{\heartsuit}$ or $\boxed{\heartsuit}\boxed{\clubsuit}$, then it implies $x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 0$.
If the resulting sequence is $\boxed{\clubsuit}\boxed{\clubsuit}$ or $\boxed{\heartsuit}\boxed{\heartsuit}$, then it implies $x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 1$.

Note that steps 2 and 3 are repeated until the type of the first card differs from the type of the fifth card. Therefore, since each deck of the four leftmost cards and the four rightmost cards consists of two $\boxed{\clubsuit}$'s and two $\boxed{\heartsuit}$'s, the protocol takes an average of two trials. Furthermore, opening the two cards in step 3 does not leak any information about the values of x_1, x_2, x_3 and x_4.

After the first and fifth cards become different types, i.e. they become $\boxed{\clubsuit}\boxed{\heartsuit}$ or $\boxed{\heartsuit}\boxed{\clubsuit}$, the third and seventh cards are opened in step 4. Hence, there are exactly 8 possibilities for the sequence of the first, third, fifth and seventh cards, as follows.

1st card	3rd card	$x_1 \oplus x_2$	5th card	7th card	$x_3 \oplus x_4$	3rd and 7th cards	$x_1 \oplus x_2 \oplus x_3 \oplus x_4$
♣	♣	0	♡	♡	0	♣ ♡	0
♡	♣	1	♣	♡	1		
♡	♡	0	♣	♣	0	♡ ♣	
♣	♡	1	♡	♣	1		
♣	♣	0	♡	♣	1	♣ ♣	1
♡	♣	1	♣	♣	0		
♡	♡	0	♣	♡	1	♡ ♡	
♣	♡	1	♡	♡	0		

Remember the observation in Section 4.2.1: the first and third cards (resp. the fifth and seventh cards) are $\boxed{\clubsuit}\boxed{\clubsuit}$ or $\boxed{\heartsuit}\boxed{\heartsuit}$ if and only if $x_1 \oplus x_2 = 0$ (resp. $x_3 \oplus x_4 = 0$). Also, remember that the first and fifth cards are different types. Therefore, the third and seventh cards are $\boxed{\clubsuit}\boxed{\heartsuit}$ or $\boxed{\heartsuit}\boxed{\clubsuit}$ if and only if $x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 0$. Thus, the protocol correctly produces the output $x_1 \oplus x_2 \oplus x_3 \oplus x_4$.

Moreover, opening the third and seventh cards does not leak any information other than just the value of $x_1 \oplus x_2 \oplus x_3 \oplus x_4$; for example, no matter whether the third and seventh cards are $\boxed{\clubsuit}\boxed{\heartsuit}$ or they are $\boxed{\heartsuit}\boxed{\clubsuit}$, all the information we gain is just the value of $x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 0$.

Thus, the protocol above securely computes $x_1 \oplus x_2 \oplus x_3 \oplus x_4$ in a non-committed format.

4.3 Complete description of our protocol

The following is the complete description of our XOR protocol, which securely computes $a \oplus b$ using 10 cards.

1. Put the 10 cards as follows:

$$\underbrace{\boxed{?}\boxed{?}}_{a}\underbrace{\boxed{?}\boxed{?}}_{b}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}.$$

2. Turn over all the face-up cards, and apply a random cut to the four rightmost cards:

$$\underbrace{\boxed{?}\boxed{?}}_{a}\underbrace{\boxed{?}\boxed{?}}_{b}\underbrace{\boxed{?}\boxed{?}}_{0}\;\left\langle\underbrace{\boxed{?}\boxed{?}}_{0}\underbrace{\boxed{?}\boxed{?}}_{0}\right\rangle$$
$$\downarrow$$
$$\underbrace{\boxed{?}\boxed{?}}_{a}\underbrace{\boxed{?}\boxed{?}}_{b}\underbrace{\boxed{?}\boxed{?}}_{0}\;\underbrace{\boxed{?}\boxed{?}}_{r}\underbrace{\boxed{?}\boxed{?}}_{r}.$$

3. Apply two random cuts as follows:

$$\left\langle\boxed{?}\boxed{?}\boxed{?}\boxed{?}\right\rangle\;\left\langle\boxed{?}\boxed{?}\boxed{?}\boxed{?}\right\rangle\;\underbrace{\boxed{?}\boxed{?}}_{r}.$$

4. Draw the first and fifth cards without opening them:

$$\overbrace{\boxed{}}^{\text{drawn}}\boxed{?}\boxed{?}\boxed{?}\;\overbrace{\boxed{}}^{\text{drawn}}\boxed{?}\boxed{?}\boxed{?}\;\underbrace{\boxed{?}\boxed{?}}_{r}. \tag{4}$$

After applying a random cut to the two drawn cards, open the two cards. If the resulting sequence is $\boxed{\clubsuit}\boxed{\heartsuit}$ or $\boxed{\heartsuit}\boxed{\clubsuit}$, then go to step 5. If the resulting sequence is $\boxed{\clubsuit}\boxed{\clubsuit}$ or $\boxed{\heartsuit}\boxed{\heartsuit}$, then return the two cards to the sequence (4) with their faces down, and go back to step 3.

5. Open the third and seventh cards:

$$\boxed{}\boxed{?}\overbrace{\boxed{?}}^{\text{open}}\boxed{?}\;\boxed{}\boxed{?}\overbrace{\boxed{?}}^{\text{open}}\boxed{?}\;\underbrace{\boxed{?}\boxed{?}}_{r}.$$

If the resulting sequence is $\boxed{\clubsuit}\boxed{\heartsuit}$ or $\boxed{\heartsuit}\boxed{\clubsuit}$, then the two rightmost cards (i.e. the commitment to r) form the output $a \oplus b$:

$$\boxed{}\boxed{}\cdots\boxed{}\;\underbrace{\boxed{?}\boxed{?}}_{a\oplus b}.$$

If the resulting sequence is $\boxed{\clubsuit}\boxed{\clubsuit}$ or $\boxed{\heartsuit}\boxed{\heartsuit}$, then reversing the order of the two rightmost cards brings the output $a \oplus b$:

$$\boxed{}\boxed{}\cdots\boxed{}\;\underbrace{\overbrace{\boxed{?}\boxed{?}}^{\rightleftharpoons}}_{r}\;\rightarrow\;\boxed{}\boxed{}\cdots\boxed{}\;\underbrace{\boxed{?}\boxed{?}}_{a\oplus b}.$$

Note that our XOR protocol above takes an average of two trials. Furthermore, after the protocol terminates, we have 8 reusable cards.

5 Conclusions

In this paper, we addressed efficient secure computation of XOR with a deck of cards, and gave a protocol by which Alice and Bob can securely compute the XOR function $a \oplus b$ using only 10 cards, where Alice has a secret bit a and Bob has a secret bit b. Our secure XOR protocol is simple, and takes an average of two trials. As indicated in Table 1, our XOR protocol is more efficient than one given by Crépeau and Kilian [4] and one applying Stiglic's AND protocol [13].

Since our XOR protocol produces its output in a committed format, it can be easily extended to a multiparty XOR protocol which securely computes the n-variable XOR function $x_1 \oplus x_2 \oplus \cdots \oplus x_n$. One can easily observe that such a protocol requires $2n + 6$ cards, provided that reusable cards are utilized.

One may notice that the 3-variable XOR function $x_1 \oplus x_2 \oplus x_3$ can be securely computed (in a committed format) using only 10 cards, because $x_1 \oplus x_2 \oplus x_3 \oplus r$ can be also securely computed in a non-committed format by the protocol given in Section 4.2.2. Using the idea, we can construct a multiparty XOR protocol which securely computes $x_1 \oplus x_2 \oplus \cdots \oplus x_n$ using only $2n + 4$ cards if $n \geq 3$.

One of the most interesting open questions is whether or not there exists a secure XOR protocol which works with less than 10 cards using random cuts. Similarly, as proposed in [13], it is a challenging open problem to prove that Stiglic's AND protocol is optimal in the sense that the number of needed cards is minimum (or to design a committed secure AND protocol which works with less than 8 cards using random cuts).

As a building block for our secure XOR protocol, we gave a method for securely computing the 4-variable XOR function in a non-committed format using 8 cards. If one could find a method for securely computing the n-variable XOR function in a non-committed format using a reasonable number of cards, then one might have a more efficient secure multiparty XOR protocol.

References

[1] M. H. Albert, R. E. L. Aldred, M. D. Atkinson, H. P. van Ditmarsch and C. C. Handley, "Safe communication for card players by combinatorial designs for two-step protocols," Australas. J. Combin. 33 (2005), 33–46.

[2] J. Balogh, J. A. Csirik, Y. Ishai and E. Kushilevitz, "Private computation using a PEZ dispenser," Theoret. Comp. Sc. 306 (2003), 69–84.

[3] B. den Boer, "More efficient match-making and satisfiability: the five card trick," Proc. EUROCRYPT '89, Lec. Notes Comp. Sc. 434 (Springer-Verlag 1990), 208–217.

[4] C. Crépeau and J. Kilian, “Discreet solitary games,” Proc. CRYPTO ’93, Lec. Notes Comp. Sc. 773 (Springer-Verlag 1994), 319–330.

[5] H. P. van Ditmarsch, W. van der Hoek, R. van der Meyden and J. Ruan, “Model checking Russian Cards,” Electronic Notes Theoret. Comp. Sc. 149, No. 2 (2006), 105–123.

[6] M. J. Fischer and R. N. Wright, “Bounds on secret key exchange using a random deal of cards,” J. Cryptology 9 (1996), 71–99.

[7] T. Mizuki, H. Shizuya and T. Nishizeki, “A complete characterization of a family of key exchange protocols,” International Journal of Information Security 1, no. 2 (2002), 131–142.

[8] T. Mizuki, H. Shizuya and T. Nishizeki, “Characterization of optimal key set protocols,” Discrete Appl. Math. 131 (2003), 213–236.

[9] T. Moran and M. Naor, “Basing cryptographic protocols on tamper-evident seals,” Proc. ICALP 2005, Lec. Notes Comp. Sc. 3580 (Springer-Verlag, 2005), 285–297.

[10] V. Niemi and A. Renvall, “Secure multiparty computations without computers,” Theoret. Comp. Sc. 191, (1998), 173–183.

[11] A. Salomaa, “Caesar and DNA. Views on cryptology,” Proc. 12th Internat. Symposium on Fundamentals of Computation Theory (FCT ’99), Lec. Notes Comp. Sc. 1684 (Springer-Verlag, 1999), 39–53.

[12] A. Salomaa, “Public-Key Cryptography (Second, Enlarged Edition),” Springer-Verlag, Berlin, Heidelberg, New York, 1996.

[13] A. Stiglic, “Computations with a deck of cards,” Theoret. Comp. Sc. 259 (2001), 671–678.

(Received 19 Sept 2005; revised 4 Apr 2006)