Card-Based Protocols for Any Boolean Function*

Takuya Nishida[1], Yu-ichi Hayashi[1], Takaaki Mizuki[2], and Hideaki Sone[2]

[1] Graduate School of Information Sciences, Tohoku University
6–3–09 Aramaki-Aza-Aoba, Aoba-ku, Sendai, Miyagi 980–8578, Japan
[2] Cyberscience Center, Tohoku University
6–3 Aramaki-Aza-Aoba, Aoba-ku, Sendai, Miyagi 980–8578, Japan
tm-paper+cardanyw[atmark]g-mail.tohoku-university.jp

Abstract. Card-based protocols that are based on a deck of physical cards achieve secure multi-party computation with information-theoretic secrecy. Using existing AND, XOR, NOT, and copy protocols, one can naively construct a secure computation protocol for any given (multivariable) Boolean function as long as there are plenty of additional cards. However, an explicit sufficient number of cards for computing any function has not been revealed thus far. In this paper, we propose a general approach to constructing an efficient protocol so that six additional cards are sufficient for any function to be securely computed. Further, we prove that two additional cards are sufficient for any symmetric function.

1 Introduction

It is known that secure multi-party computation (MPC) can be achieved using a number of physical cards such as black $\boxed{\clubsuit}$ and red $\boxed{\heartsuit}$ cards (with identical backs $\boxed{?}$). Several card-based cryptographic protocols have been reported in the literature: in addition to the elementary computations, namely, the AND [1, 3, 7, 10, 12, 15] and XOR [3, 10, 11] protocols, *efficient* protocols (i.e., protocols that require fewer cards) have been designed for specific functions such as the adder [6] and the 3-variable functions [13]. Whereas previous studies have dealt with specific functions, this paper proposes a general approach to constructing an efficient protocol for any given (multivariable) Boolean function.

We start with introducing some preliminary notations for card-based protocols.

1.1 Preliminary Notations

To deal with Boolean values, we use the following encoding rule based on the order of a pair of cards:

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

* This paper appears in Proceedings of TAMC 2015. The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-17142-5_11.

For a bit $x \in \{0, 1\}$, a pair of face-down cards $\boxed{?}\boxed{?}$ that has a value equaling x according to encoding rule (1) is called a *commitment* to x and is written as

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

"Committed-format" protocols [3, 6, 10–13, 15] produce the output as a commitment; for example, given commitments to bits a and b, we can obtain a commitment

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

as the output of an AND or XOR protocol.

Given a pair of bits (x, y), we define two operations, get and shift, as

$$\begin{aligned} \mathsf{get}^0(x, y) &= x\ , \quad \mathsf{get}^1(x, y) = y; \\ \mathsf{shift}^0(x, y) &= (x, y)\ , \quad \mathsf{shift}^1(x, y) = (y, x). \end{aligned}$$

Using these operations, the AND function can be written as

$$a \wedge b = \mathsf{get}^{a \oplus r}(\mathsf{shift}^r(0, b)) \tag{2}$$

for an arbitrary bit $r \in \{0, 1\}$ [13]. Hereafter, for two bits x and y, the notation (i) below implies (ii).

$$\text{(i)}\ \underbrace{\boxed{?}\boxed{?}\boxed{?}\boxed{?}}_{(x,y)}, \qquad \text{(ii)}\ \underbrace{\boxed{?}\boxed{?}}_{x}\underbrace{\boxed{?}\boxed{?}}_{y}.$$

1.2 AND Protocol

Next, we introduce the most efficient AND protocol [10] currently known. Given commitments to bits a and b together with two additional cards, it achieves a committed-format AND computation as follows.

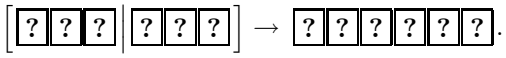
1. Arrange three commitments to a, 0, and b:

$$\underbrace{\boxed{?}\boxed{?}}_{a}\boxed{\clubsuit}\boxed{\heartsuit}\underbrace{\boxed{?}\boxed{?}}_{b} \rightarrow \underbrace{\boxed{?}\boxed{?}}_{a}\underbrace{\boxed{?}\boxed{?}}_{0}\underbrace{\boxed{?}\boxed{?}}_{b}.$$

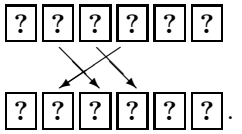
2. Rearrange the sequence of six cards as

$$\begin{array}{c} \boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?} \\ \searrow\ \swarrow\ \searrow\ \swarrow \\ \boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}\boxed{?}. \end{array}$$

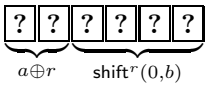
3. Bisect the sequence of six cards and switch the two portions (each of which consists of three cards) randomly; we call this a *random bisection cut* [10] and denote it by $[\cdot\,|\,\cdot]$:



4. Rearrange the sequence as

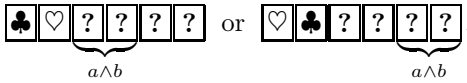


Then, we have



where r is a (uniformly distributed) random bit because of the random bisection cut.

5. Reveal the two left-most cards; then, the value of $a \oplus r$ along with Eq. (2) gives us the position of the desired commitment to $a \wedge b$:

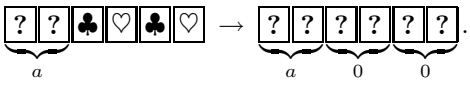


Since r is random, revealing the commitment to $a\oplus r$ does not cause any information about bit a to be leaked, and hence, this protocol achieves an information-theoretically secure computation.[1] Note that the two revealed cards can be used for another computation (we call such an available card a *free* card).

1.3 Copy Protocol

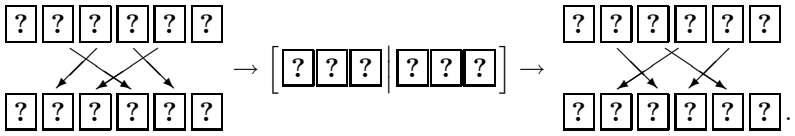
Given a commitment to bit a together with four additional cards, we can make two copied commitments to a [10] as follows.

1. Arrange three commitments to a, 0, and 0.

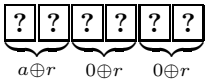


[1] Security is dependent on physical properties such as cards of the same color being indistinguishable and a random bisection cut being applied *truly randomly*. A formal treatment appears in [8], and the settings of this study are based on the formalization of card-based protocols. It is also known that one can practically assume a semi-honest model, i.e., a protocol is always executed properly [9].

2. Rearrange the sequence, apply a random bisection cut, and rearrange it again:

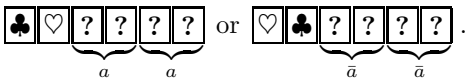


Then, we have



where r is a random bit.

3. Reveal the two left-most cards; then, we know whether $r = a$ or $r = \bar{a}$, and we have

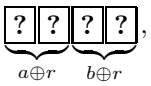


Hence we obtain two commitments to a.

Note that swapping the two cards that constitute a commitment to a bit results in a commitment to the negation of the bit (recall encoding rule (1)), i.e., the NOT computation is trivial. Therefore, hereafter, we omit detailed descriptions of how a commitment to negation $\bar{x}$ is transformed into a commitment to x.

If we start this protocol with commitments to a, b, and 0 in step 1 instead, commitments to $a \oplus b$ and a will be obtained [13].

Similarly, given commitments to bits a and b, we easily obtain



and hence the existing XOR protocol [10] produces a commitment to $a \oplus b$ without the use of any additional card.

1.4 Our Results

The existing AND, XOR, and NOT protocols introduced thus far immediately imply the following theorem.

Theorem 1 ([10]). *Given commitments to x_1 and x_2 together with two additional cards*  *, we can securely produce a commitment to the value of any 2-variable Boolean function $f(x_1, x_2)$.*

It is also known that the following holds.

Theorem 2 ([13]). *Given commitments to x_1, x_2, x_3 together with two additional cards* *, we can securely produce a commitment to the value of any 3-variable Boolean function $f(x_1, x_2, x_3)$.*

These two results raise a natural question: what about the case of any general Boolean function having four or more variables? Of course, by combining the existing AND, XOR, NOT, and copy protocols, one can securely compute any (multivariable) Boolean function $f(x_1, x_2, \ldots, x_n)$ as long as there are plenty of additional cards. However, an explicit sufficient number of cards for computing any function has not been revealed thus far. We investigate this open problem and propose a general approach to constructing an efficient protocol, showing sufficient conditions on the numbers of additional cards.

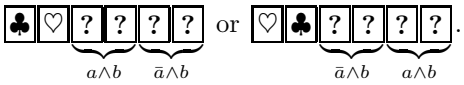
The remainder of this paper is organized as follows. In Section 2, we improve the existing AND and half-adder protocols. In Section 3, using the improved AND protocol, we demonstrate the construction of a protocol that securely computes any given n-variable Boolean function with n input commitments and six additional cards, i.e., we prove that six additional cards are sufficient for this case. In Section 4, using our improved half-adder protocol, we show that two additional cards are sufficient for the case of symmetric functions. Finally, the paper is concluded in Section 5.

2 Building Blocks

In this section, we create two new protocols as building blocks for the main results (presented in Sections 3 and 4) by modifying the known AND protocol [10] introduced in Section 1.2. The first new protocol produces a commitment to $a \wedge b$ as well as a commitment to b, as described in Section 2.1. The second one achieves half-adder computation with only two additional cards, as described in Section 2.2.

2.1 Improved AND Protocol

Recall the AND protocol [10] introduced in Section 1.2. When a commitment to $a \wedge b$ is obtained as the output of the protocol, the other two face-down cards will constitute a commitment to $\bar{a} \wedge b$, as known from Eq. (2):

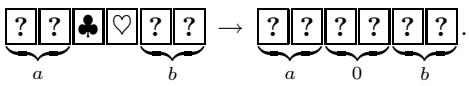


From this observation and the identity

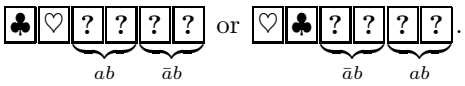
$$ab \oplus \bar{a}b = (a \oplus \bar{a})\, b = b$$

(where we omit the conjunction symbol $\wedge$ hereafter), we can improve the AND protocol so that one of the input commitments will be retained, as follows.

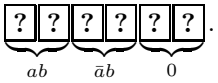
1. Arrange three commitments to a, 0, and b.



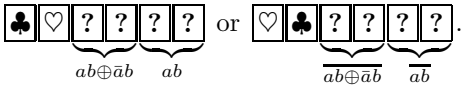
2. Apply the known AND protocol [10]; then, we have



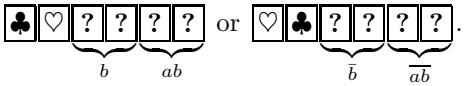
3. Rearrange the sequence as



4. Apply steps 2 and 3 of the copy protocol [10] introduced in Section 1.3; then, we have



Since $ab \oplus \bar{a}b = b$, we have



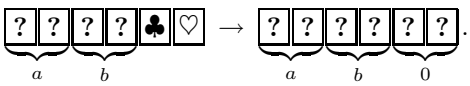
Thus, this protocol allows us to retain a commitment to b. Therefore, the following lemma holds.

Lemma 3. *Given commitments to x_1 and x_2 together with two additional cards* ♣♡*, we can securely produce commitments to $x_1 x_2$ and x_2.*

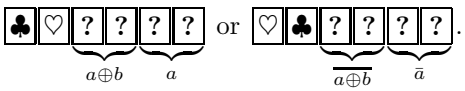
2.2 Improved Half-Adder Protocol

It is known that half-adder computation can be achieved with eight cards [6], i.e., given commitments to a and b together with four additional cards, the existing protocol produces commitments to $a \oplus b$ and ab. In this subsection, we improve the half-adder protocol by applying the improved AND protocol described in the previous subsection. Our half-adder protocol requires only two additional cards and proceeds as follows.

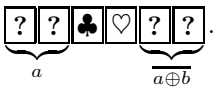
1. Arrange three commitments to a, b, and 0.



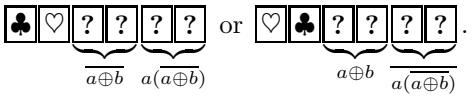
2. Apply steps 2 and 3 of the copy protocol [10] introduced in Section 1.3; then, we have



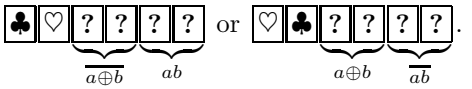
3. Rearrange the sequence as



4. Apply the improved AND protocol described in the previous subsection; then, we have



Since $a\,(\overline{a \oplus b}) = a\bar{a} \oplus ab = ab$, we have



Thus, this protocol achieves half-adder computation using only two additional cards. Therefore, the following lemma holds.

Lemma 4. *Given commitments to x_1 and x_2 together with two additional cards* ♣♡*, we can securely produce commitments to $x_1 \oplus x_2$ and x_1x_2.*

3 Computation of Any Multivariable Function

In this section, we present a general approach to constructing an efficient protocol for any given n-variable Boolean function by showing that any n-variable function can be securely computed with n input commitments and six additional cards.

3.1 Concepts and Sub-Protocol

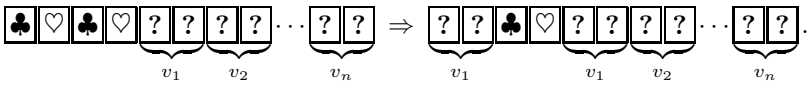
Remember that XOR computation can be easily achieved [10] as described in Section 1.3. Hence, XOR computation should be employed to construct an efficient protocol. Therefore, we consider AND-XOR expressions of a given function. Indeed, it is well known that any n-variable function $f(x_1, x_2, \ldots, x_n)$ can be expressed as the Shannon expansion (or Boole's expansion) [14]:

$$\begin{aligned} f(x_1, x_2, \ldots, x_n) = {} & \bar{x}_1\bar{x}_2\cdots\bar{x}_n f(0,0,\ldots,0) \oplus x_1\bar{x}_2\cdots\bar{x}_n f(1,0,\ldots,0) \\ & \oplus \bar{x}_1 x_2\cdots\bar{x}_n f(0,1,\ldots,0) \oplus x_1x_2\cdots\bar{x}_n f(1,1,\ldots,0) \\ & \oplus \cdots \oplus x_1x_2\cdots x_n f(1,1,\ldots,1). \end{aligned}$$

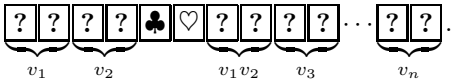
i.e., $f(x_1, x_2, \ldots, x_n)$ can be expressed uniquely by combining 2^n product terms with XORs, where a product term can be deleted if the corresponding value of f is 0.

Now, we want to handle product terms $v_1v_2\cdots v_n$, where v_i, $1 \le i \le n$, is a literal (either x_i or $\bar{x}_i$): given commitments to $v_1, v_2, \ldots, v_n$ together with four additional cards, the following sub-protocol securely generates a commitment to the product term $v_1v_2\cdots v_n$.

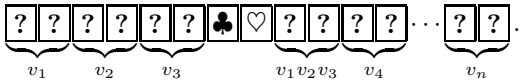
1. Make two copied commitments to v_1 using the copy protocol [10] introduced in Section 1.3:



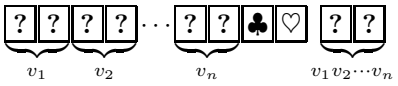
2. Apply the AND protocol described in Section 2.1; then, by Lemma 3 we have



3. Similarly, by Lemma 3 we have



4. Repeat this up to v_n so that we have



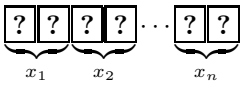
Thus, four additional cards allow us to generate a commitment to the product term without losing the input commitments.

Lemma 5. *Given commitments to literals $v_1, v_2, \ldots, v_n$ together with four additional cards* ♣♣♡♡*, we can securely produce commitments to $v_1, v_2, \ldots, v_n$, and $v_1 v_2 \cdots v_n$.*

3.2 Complete Description of Protocol

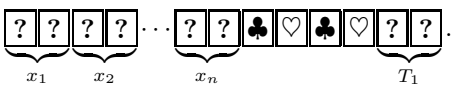
Now, we are ready to present our general protocol for securely computing any function.

Let f be an arbitrary n-variable function. Given n commitments

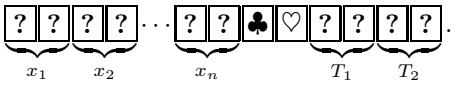


and six additional cards ♣♣♣♡♡♡, the following protocol securely produces a commitment to $f(x_1, x_2, \ldots, x_n)$.

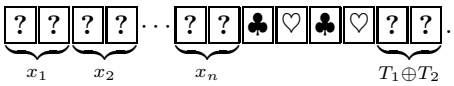
1. Let $T_1 \oplus T_2 \oplus \cdots \oplus T_\ell$ be the Shannon expansion of f after removing the constant-zero terms (where T_i, $1 \leq i \leq \ell$, is a product term). Generate a commitment to T_1 using the sub-protocol described in Section 3.1. Then, by Lemma 5 we have



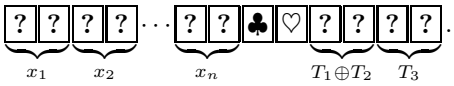
2. Generate a commitment to T_2 by Lemma 5:



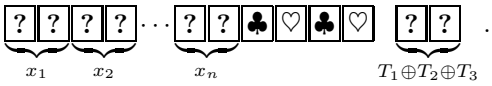
3. Apply the XOR protocol [10] to the two right-most commitments:



4. Generate a commitment to T_3 by Lemma 5:



5. Apply the XOR protocol [10] to the two right-most commitments:



6. Repeat this until we get a commitment to $T_1 \oplus T_2 \oplus \cdots \oplus T_\ell$, which is equal to $f(x_1, x_2, \ldots, x_n)$:

The remaining commitments to $x_1, x_2, \ldots, x_n$ as well as the four free cards ♣♣♡♡ can be used for another computation. Thus, we have the following theorem.

Theorem 6. *Let f be an n-variable function. Given commitments to $x_1, x_2, \ldots, x_n$ together with six additional cards ♣♣♣♡♡♡, we can securely produce commitments to $x_1, x_2, \ldots, x_n$, and the value $f(x_1, x_2, \ldots, x_n)$.*

Note that our improved AND protocol (Lemma 3) plays an important role in reducing the number of required cards; without it, two more cards would be required to run the sub-protocol, and consequently, the above protocol. Furthermore, although we used the Shannon expansion in step 1, one may use any AND-XOR expression instead, which can be obtained by applying some simplification algorithm [14].

4 Case of Symmetric Functions

The previous section described the construction of a protocol that securely produces a commitment to the value of any function using n input commitments and six additional cards. In this section, we focus our attention on symmetric functions because practically important functions in MPC are often symmetric.

Specifically, we prove that two additional cards are sufficient for the case of symmetric functions.

Let f be an n-variable symmetric function. Then, the value $f(x_1, x_2, \ldots, x_n)$ depends on only the number of variables that take 1, namely $\sum_{i=1}^{n} x_i$. For instance, the 3-input majority MAJ_3, which is a 3-variable symmetric function, can be expressed using a function $g : \{0, 1, 2, 3\} \to \{0, 1\}$ as

$$\mathsf{MAJ}_3(x_1, x_2, x_3) = g\left(\sum x_i\right) = \begin{cases} 0 & \text{if } \sum x_i \leq 1 \\ 1 & \text{otherwise.} \end{cases}$$

Thus, for any n-variable symmetric function $f : \{0, 1\}^n \to \{0, 1\}$, there exists a unique function $g : \{0, 1, \ldots, n\} \to \{0, 1\}$ such that $f(x_1, x_2, \ldots, x_n) = g\left(\sum x_i\right)$.

Given commitments to $x_1, x_2, \ldots, x_n$, it is obvious that the half-adder computation described in Section 2.2 enables us to securely generate a sequence of commitments corresponding to the binary representation of $\sum x_i$; Lemma 4 implies that two additional cards are sufficient for this purpose.

Lemma 7. *Given commitments to $x_1, x_2, \ldots, x_n$ together with two additional cards* $\boxed{\clubsuit}\boxed{\heartsuit}$*, we can securely produce a $(\lfloor \log_2 n \rfloor + 1)$-bit sequence of commitments corresponding to $\sum_{i=1}^{n} x_i$.*

Note that after generating a sequence of commitments to $\sum x_i$ from commitments to $x_1, x_2, \ldots, x_n$ (and two additional cards), some free cards will arise; more specifically, we will have a total of $2\left(n - \lfloor \log_2 n \rfloor\right)$ free cards.

Now, we are ready to present our main result of this section, i.e., only two additional cards are sufficient for the case of symmetric functions.

Theorem 8. *Let $n \geq 4$ and let f be an n-variable symmetric function. Given commitments to $x_1, x_2, \ldots, x_n$ together with two additional cards* $\boxed{\clubsuit}\boxed{\heartsuit}$*, we can securely produce a commitment to the value $f(x_1, x_2, \ldots, x_n)$.*

Proof. Let $g : \{0, 1, \ldots, n\} \to \{0, 1\}$ be the function such that $g\left(\sum x_i\right) = f(x_1, x_2, \ldots, x_n)$. By Lemma 7, we obtain a $(\lfloor \log_2 n \rfloor + 1)$-bit sequence of commitments corresponding to $\sum x_i$ and $2\left(n - \lfloor \log_2 n \rfloor\right)$ free cards. If $n \geq 5$, then $2\left(n - \lfloor \log_2 n \rfloor\right) \geq 6$, and hence there are at least 6 free cards, and consequently, we can securely generate a commitment to the value $g\left(\sum x_i\right)$ by Theorem 6 (by regarding the domain of g as $\{0, 1\}^{\lfloor \log_2 n \rfloor + 1}$). Similarly, if $n = 4$, then there are a 3-bit sequence of commitments and 4 free cards, and hence Theorem 2 completes the proof. □

5 Conclusion

We proposed a general approach to designing an efficient card-based protocol for any given function. Specifically, using two-level AND-XOR representations, we can construct a protocol that requires only six additional cards to securely produce a commitment to the value of any n-variable function, regardless of how large n is (Theorem 6). Further, we showed that two additional cards are sufficient for the case of symmetric functions (Theorem 8).

As mentioned above, six additional cards are sufficient for general functions, and two additional cards are sufficient for symmetric functions. Determining whether they are necessary is an open problem; for example, is there a symmetric function that needs at least two additional cards? Note that to prove such a lower bound, one has to follow the formal computational model for card-based protocols [8].

Cryptography and playing cards share a deep connection (e.g., [2, 4, 5, 16]). One benefit of considering such a connection is that it enables us to easily demonstrate the underlying concepts of MPC and cryptography to non-specialists. In addition, we have already confirmed that ordinary people such as high-school students can use card-based protocols in their daily activities.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 26330001.

References

1. den Boer, B.: More efficient match-making and satisfiability: the five card trick. In: Quisquater, J.J., Vandewalle, J. (eds.) Advances in Cryptology — EUROCRYPT '89, Lecture Notes in Computer Science, vol. 434, pp. 208–217. Springer Berlin Heidelberg (1990)
2. Cordón-Franco, A., Van Ditmarsch, H., Fernández-Duque, D., Soler-Toscano, F.: A colouring protocol for the generalized Russian cards problem. Theoretical Computer Science 495, 81–95 (2013)
3. Crépeau, C., Kilian, J.: Discreet solitary games. In: Stinson, D.R. (ed.) Advances in Cryptology — CRYPTO '93, Lecture Notes in Computer Science, vol. 773, pp. 319–330. Springer Berlin Heidelberg (1994)
4. Duan, Z., Yang, C.: Unconditional secure communication: a Russian cards protocol. Journal of Combinatorial Optimization 19(4), 501–530 (2010)
5. Fischer, M.J., Wright, R.N.: Bounds on secret key exchange using a random deal of cards. Journal of Cryptology 9(2), 71–99 (1996)
6. Mizuki, T., Asiedu, I.K., Sone, H.: Voting with a logarithmic number of cards. In: Mauri, G., Dennunzio, A., Manzoni, L., Porreca, A.E. (eds.) Unconventional Computation and Natural Computation, Lecture Notes in Computer Science, vol. 7956, pp. 162–173. Springer Berlin Heidelberg (2013)
7. Mizuki, T., Kumamoto, M., Sone, H.: The five-card trick can be done with four cards. In: Wang, X., Sako, K. (eds.) Advances in Cryptology — ASIACRYPT 2012, Lecture Notes in Computer Science, vol. 7658, pp. 598–606. Springer Berlin Heidelberg (2012)
8. Mizuki, T., Shizuya, H.: A formalization of card-based cryptographic protocols via abstract machine. International Journal of Information Security 13(1), 15–23 (2014)
9. Mizuki, T., Shizuya, H.: Practical card-based cryptography. In: Ferro, A., Luccio, F., Widmayer, P. (eds.) Fun with Algorithms, Lecture Notes in Computer Science, vol. 8496, pp. 313–324. Springer International Publishing (2014)

10. Mizuki, T., Sone, H.: Six-card secure AND and four-card secure XOR. In: Deng, X., Hopcroft, J.E., Xue, J. (eds.) Frontiers in Algorithmics, Lecture Notes in Computer Science, vol. 5598, pp. 358–369. Springer Berlin Heidelberg (2009)
11. Mizuki, T., Uchiike, F., Sone, H.: Securely computing XOR with 10 cards. The Australasian Journal of Combinatorics 36, 279–293 (2006)
12. Niemi, V., Renvall, A.: Secure multiparty computations without computers. Theoretical Computer Science 191(1–2), 173–183 (1998)
13. Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Securely computing three-input functions with eight cards. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E98-A(6) (2015), to appear.
14. Sasao, T.: Switching Theory for Logic Synthesis. Kluwer Academic Publishers, Norwell, MA, USA, 1st edn. (1999)
15. Stiglic, A.: Computations with a deck of cards. Theoretical Computer Science 259(1–2), 671–678 (2001)
16. Swanson, C.M., Stinson, D.R.: Combinatorial solutions providing improved security for the generalized Russian cards problem. Designs, Codes and Cryptography 72(2), 345–367 (2014)