Computations with a deck of cards

Anton Stiglic
Zero-Knowledge Systems Inc
888 de Maisonneuve East, 6th Floor
Montréal (Québec), Canada
e-mail: anton@zeroknowledge.com

Abstract

A deck of cards can be used as a cryptographic tool ([3], [6]). Using a protocol that securely computes the Boolean AND function, one can construct a protocol for securely computing any Boolean function. This, in turn, can be used for secure multiparty computations, solitary games, zero knowledge proofs and other cryptographic schemes.

We present a protocol for 2 people to securely compute the AND function using a deck of 2 types of cards. The protocol needs a total of only 8 cards, thus confirming the assumption of an open question [3] about the minimal number of values that are needed for this type of computation. To our knowledge, the protocol is also the first one of its kind that does not need to make copies of the inputs. We thus prove upper bounds for this type of computation. The protocol is much simpler, uses less cards, and is more efficient than the ones introduced in [3] and [6].

1 Introduction

Suppose $Alice$ commits herself to a bit b_A and Bob commits himself to b_B. We would like $Alice$ and Bob to be able to compute $b_A \wedge b_B$ in such a way that neither one of them learns anything more than what they can deduce from their own input and the output of the computation (for example, if $Alice$ is committed to 0, she will never know what bit Bob was committed to). Bert den Boer [4] first introduced a now classic protocol that enables two participants to privately compute the AND function of their inputs. To be able to compute any Boolean function (see section 6) it is necessary that the answer be in a committed format. Claude Crépeau and Joe Kilian came up with a solution to this problem in [3], using 4 types of cards. Later on, Valtteri Niemi and Ari Renvall proposed a solution in [6] that used only 2 types of cards. Although our solution is only linearly more efficient than the latter one (which in turn, is only linearly more efficient than the one in [3]), it proves important upper bounds and may be the most simple and efficient one that exists. A protocol for securely computing the Boolean AND function is an important cryptographic tool with many applications, it can be used for multiparty computations, solitary games, zero-knowledge proofs and more (we discuss these later

on, see also [4], [3], [1], [6]). Although the number of cards needed for the computation of a Boolean function increases only linearly with the number of gates of the circuit defining it, complex computations demand an extremely large amount of cards. Only small computations of these kind can be done efficiently with cards, thus even slight optimizations of the AND protocol is useful.

2 The Model

We will be working with the following alphabet:

$$\Sigma = \{ \boxed{\heartsuit}, \boxed{\clubsuit}, \boxed{?} \}$$

Each value can be thought of as a suit in a deck of cards, $\boxed{?}$ representing a card with it's face down.
Let $c_1, c_2, \ldots, c_n$ be elements of Σ. $c_1 c_2 \ldots c_n$ can be considered as a deck of cards, c_1 being the topmost card, c_2 the second, etc...
We define $(c_1 c_2 \ldots c_n)$ as the set $\{c_1 c_2 \ldots c_n,\ c_2 c_3 \ldots c_n c_1,\ \ldots,\ c_n c_1 \ldots c_{n-1}\}$ (i.e the set of cyclic permutations of letters of the string $c_1 c_2 \ldots c_n$).
$\langle \cdot \rangle$ will denote the operator that takes an element from the set Σ^* to the set Σ^* such that

$$\langle c_1, c_2, \ldots, c_n \rangle \longrightarrow \pi$$

where π is picked randomly in $(c_1 c_2 \ldots c_n)$.
Applying $\langle \cdot \rangle$ to a string can be thought of as applying a cyclic shuffling of the cards represented by the string.
We will use the following coding:

$$\boxed{\heartsuit}\boxed{\clubsuit} = 1,\ \boxed{\clubsuit}\boxed{\heartsuit} = 0$$

e will be a function which corresponds to turning a "string" of cards face down and δ will be the inverse of e. We suppose that we cannot distinguish between $\boxed{\heartsuit}\boxed{\clubsuit}$ and $\boxed{\clubsuit}\boxed{\heartsuit}$ when they are face down $\boxed{?}\boxed{?}$ and once we have applied $\langle \cdot \rangle$ to them.

3 Bit Commitment Protocol

Say $Alice$ wants to commit to a bit b, she simply does the following

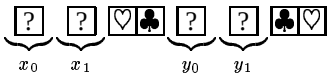
1. She takes two distinct cards $\boxed{\heartsuit}\boxed{\clubsuit}$, shows them to Bob and then places them face down $\boxed{?}\boxed{?}$ (she applies e). Call this string α.
2. She then computes $\alpha' := \langle \alpha \rangle$.
3. She outputs α'.

To reveal the secret, we simply compute $\delta(\alpha')$ (i.e., we turn over the cards).

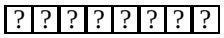
4 Secure AND protocol

[4] first proposed a protocol to securely compute $b_A \wedge b_B$ but the result was not in a committed format. Claude Crépeau and Joe Kilian proposed a *Las Vegas* algorithm in [3] that produced a committed output but it uses a larger alphabet (a deck of 4 different types of cards), needs to make copies of the cards that commit the input and takes an average of 12 trials. Valtteri Niemi and Ari Renvall also proposed a solution in [6], their *Las Vegas* algorithm used only 2 types of cards but also needed to make copies of the input, took an average of 2.5 trials and the AND protocol needed a total of 10 cards. The algorithm proposed here uses 2 types of cards and takes an average of 2 trials, no copies of the committed inputs are needed and the total number of cards needed is just 8. This gives an upper bound to the number of values (4 values coded by 8 cards) needed to be shuffled during the AND protocol, proving the assumption in the open question of [3]. It also gives an upper bound to the number of copies needed of the inputs: NO copies of the inputs need to be made.
Our protocol works as follows:
Denote $x_0 x_1$ as the cards that commit $Alice$s value b_A and $y_0 y_1$ the cards that commit Bobs value b_B. These cards are of the form $\boxed{?}\boxed{?}$, turned over they are either $\boxed{\heartsuit}\boxed{\clubsuit}$ or $\boxed{\clubsuit}\boxed{\heartsuit}$. We need 4 extra cards: 2 $\boxed{\heartsuit}$'s and 2 $\boxed{\clubsuit}$'s.

1. Place the cards as follows



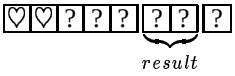
2. Then turn over the public cards, let's call this "string" ω.



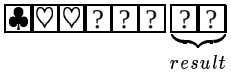
3. Now let $Alice$ and then Bob apply a cyclic shuffling: $\tilde{\omega} \leftarrow \langle \omega \rangle$.

4. Turn over the two topmost cards of $\tilde{\omega}$, call this v.
   If $v \in \{\boxed{\heartsuit}\boxed{\heartsuit}, \boxed{\clubsuit}\boxed{\clubsuit}\}$ then go on to step $(5.)$.
   If $v = \boxed{\clubsuit}\boxed{\heartsuit}$, then turn over the third topmost card, if it is a $\boxed{\heartsuit}$, go on to the next step, otherwise turn back over the public cards and go back to the cyclic shuffling step $(3.)$.
   If $v = \boxed{\heartsuit}\boxed{\clubsuit}$, then turn over the third topmost card, if it is a $\boxed{\clubsuit}$, go on to the next step, otherwise turn back over the cards and go back to step $(3.)$.

5. If the 2 topmost cards are $\boxed{\heartsuit}\boxed{\heartsuit}$, then the 6th and 7th topmost cards are the commitment to the result



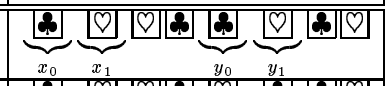
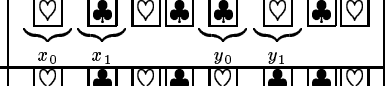
   If the 3 topmost cards are $\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}$, then the $7th$ and $8th$ cards are the commitment to the result



If the 2 topmost cards are ♣♣, then the $4th$ and $5th$ cards contain the commitment to the result

Finally, if the 3 topmost cards are ♡♣♣, then the $5th$ and $6th$ cards contain the commitment to the result

To see why the protocol works and is secure, let's see what happens from "under the glass table": At step two, we get one of the following configurations

b_A	b_B	ω uncovered
0	0	
0	1	
1	0	
1	1	

$\tilde{\omega}$ is just one of the above card configurations permuted by a cyclic shift, this is just done so that Bob and $Alice$ have no information on the order of the cards and the act of turning the topmost card becomes equivalent to picking, uniformly at random, a card from the deck. Now, after the cyclic shuffling, the probability that the 2 topmost cards are ♡♡ is $\frac{1}{8}$, in all 4 cases and the probability that they are ♣♣ is also $\frac{1}{8}$ in all 4 cases, so we get absolutely no information on the inputs of $Alice$ and Bob. On the other hand, the probability of picking ♡♣ is $\frac{3}{8}$ in all 4 cases, same thing for picking ♣♡, so no information is leaked here either.
Finally, if we picked ♡♣, the probability of picking a ♡ as the third card is $\frac{2}{3}$, and the probability of picking a ♣ for a third card is $\frac{1}{3}$, in *all* 4 cases. The probability of picking ♡♣♣ or ♣♡♡ is also equiprobable in all four cases. These are all the situations we will encounter, all probabilities are equiprobable in all four cases, thus demonstrating that our protocol is secure.

The fact that the protocol gives the commitment to the right answer can easily be seen by observing the value coded by the cards to be picked by the protocol.

5 Other primitives

In order to be able to privately compute any probabilistic Boolean function we first need to describe a few more primitives.

5.1 OR, NOT gates

It is easy to compute the negation of a committed bit, you simply reverse the order of the two cards. With this in hand, and the AND protocol described in section 4, we can easily construct a protocol for the OR gate ($b_A \vee b_B \equiv \neg b_A \wedge \neg b_B$).

5.2 Random committed bits

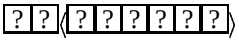
For a probabilistic Boolean function, we can get random bits by taking cards committing bits and applying $\langle \cdot \rangle$ to them.

5.3 Copies of a committed bit

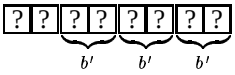
Although copies of the committed bits are not need to compute a simple boolean gate, it is a tool that is needed for privately computing any Boolean function. We present a protocol that enables us to make n copies of a committed bit, for any n. The protocol comes directly from [3]
To copy a committed bit b:

1. create the following configuration

2. Turn over the public cards, and apply a random cyclic shift to the 6 rightmost cards

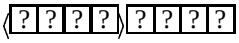


   We get the following configuration

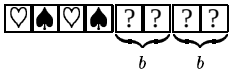


   where b' is now an unknown bit

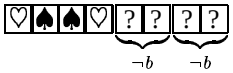
3. Now randomly shift the 4 topmost values



4. Open the 4 topmost values.
   If the sequence you see is alternating then it means that $b = b'$ and the 4 rightmost cards form 2 copies of b.



   Otherwise, the 4 rightmost values form 2 copies of $\neg b$



This protocol is easily generalized to make any number (n) of copies.

6 Computations with cards

6.1 Multi-Party Computations

The notion of multiparty computation (MPC) was first introduced in [7]. A first protocol permitting a general multiparty computation, as well as completeness theorems, was given in [5]. The MPC problem can be defined as follows: a group of n players $P_1, \ldots P_n$ wish to securely (and correctly) compute $F(x_1, \ldots, x_n)$, where x_i is P_i's private input and F is a public function which they have agreed upon. Securely here means that a player p_i does not get to know any more information than what he can deduce from his own input and the result of the function. We assume here that the participants always follow the protocol, in an other case a more specific definition of security must be provided (see [5] and [2] for example). Also, if a group of participants decide to collide together, they must form a *minority* of the total number of participants.

As mentioned in [3] and [6], we can use the tools presented here to enable multiparty computations of any Boolean function. We simply publicly describe a Boolean circuit (AND, OR and NOT gates) defining the function and, using protocols described above, securely compute each gate, keeping the answers in committed format and using them for other inputs when necessary. The inputs of the participants are of course introduced in a committed format. Only the final answer of the function is revealed. Probabilistic Boolean functions can also be securely computed using the protocol described to generate random committed bits.

6.2 Perfect Zero Knowledge Proofs

A Zero Knowledge Proof (ZKP) consists of an all powerful prover P and a polynomial timed bounded verifier V. P would like to convince V that he possesses an answer to a certain problem without giving him the solution. We can use our protocol to construct a ZKP for any NP-COMPLETE decision problem. Simply reduce the problem to the SAT problem (call the formula f). Now P, having the solution, commits to the bits that satisfy f and securely computes f with these inputs. P reveals the final answer to V. All of this is done in polynomial time, so V can verify.

6.3 Solitary Games

As discussed in [3], any game can be played solitarily by describing the strategies of one's opponents in a probabilistic boolean circuit. POKER and BRIDGE are such examples. To play in solitary one discreetly applies the strategies of the opponents by using the secure protocols described above.

7 Remarks and Open Questions

1. We assumed that cyclic permutations (*cyclic shufflings*) of a deck of cards are indistinguishable. A question that remains open is if there are more general primitives that may allow us to do the same computations as discussed in this

paper (for example, [6] suggested to try moving from a cyclic symmetry group to a dihedral group).

2. A proof that the result presented in this paper, working in cyclic groups, is optimal concerning the amount of cards that need to be used would be good. We have started such proofs under certain conditions (no copying, 2 types of cards, using the commitment scheme described in this paper), but a more generalized proof would be better.

7.1 Achnowledgement

We would like to thank Alain Tapp, Niel Stewart, Frédéric Légaré and Adam Smith for their appreciated comments concerning earlier versions of this paper. We would also like to thank the anonymous referee for some final corrections.

References

[1] David Chaum, Ivan B. Damgård, and Jeroen van de Graaf. Multiparty computations ensuring privacy of each party's input and correctness of the result. In Carl Pomerance, editor, *Proc. CRYPTO 87*, pages 87–119. Springer-Verlag, 1988. Lecture Notes in Computer Science No. 293.

[2] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations with dishonest minority. In *Advances in Cryptology—EUROCRYPT 99*, volume 1561 of *Lecture Notes in Computer Science*, pages 311–326. Springer-Verlag, March 1999.

[3] Claude Crépeau and Joe Kilian. Discreet solitary games. In Douglas R. Stinson, editor, *Advances in Cryptology: CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 319–330. Springer, 1994.

[4] Bert den Boer. More efficient match-making and satisfiability: The five card trick. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology—EUROCRYPT 89*, volume 434 of *Lecture Notes in Computer Science*, pages 208–217. Springer-Verlag, 1990, 10–13 April 1989.

[5] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game — A completeness theorem for protocols with honest majority. In ACM, editor, *Proceedings of the nineteenth annual ACM Symposium on Theory of Computing, New York City, May 25–27, 1987*, pages 218–229, New York, NY 10036, USA, 1987. ACM Press.

[6] Valtteri Niemi and Ari Renvall. Secure multiparty computations without computers. *Theoretical Computer Science*, 191(1–2):173–183, 30 January 1998. Note.

[7] A. Yao. Protocols for secure computation. In IEEE, editor, *23rd annual Symposium on Foundations of Computer Science, November 3–5, 1982, Chicago, IL*, pages 160–164, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1982. IEEE Computer Society Press.