This paper will appear in "Advances in Cryptology: Proceedings of Crypto '93", Springer

Discreet Solitary Games

Claude Crépeau[1]* and Joe Kilian[2]

[1] LIENS (CNRS URA 1327),
45 rue d'Ulm, 75230 Paris CEDEX 05, France.
e-mail: crepeau@dmi.ens.fr.
[2] NEC Research Institute,
4 Independence Way, Room 1C01,
South Brunswick, NJ 08540, USA.
e-mail: joe@research.nj.nec.com.

Abstract. Cryptographic techniques have been used intensively in the past to show how to play multiparty games in an adversarial scenario. We now investigate the cryptographic power of a deck of cards in a solitary scenario. In particular, we show how a person can select a random permutation satisfying a certain criterion *discreetly* (without knowing which one was picked) using a simple deck of cards. We also show how it is possible using cards to play games of partial information such as POKER, BRIDGE and other cards games in solitary.

1 Introduction

It's nearly Christmas time and you have to buy presents for your family and friends. Indeed, among certain families there is a more economical approach to this situation than buying one present per person: each member of a group picks the name of another member and becomes responsible for buying that person a present. That way everybody gets something but each person buys a single present. Traditionally, the one person each member is responsible for is allocated at random using the "names-in-the-hat" technique: each person puts its name in a common hat and then everybody picks a name at random from the hat. If by accident one picks his own name, he puts it back, otherwise he is responsible for the present of the person he picked. To put it abstractly, the goal is for the n persons involved to pick a random permutation π in a way that each of them p_i knows nothing but $\pi(i)$.

Indeed the "names-in-the-hat" technique leaks some information since participant p_i who picks his own name learns that $p_1, ..., p_{i-1}$ did not pick his name. In order for this technique not to leak information whatsoever, it is necessary to start from scratch each time someone picks his own name. One can check fairly easily that a random permutation of n elements will have no fixed point with probability roughly $\frac{1}{e}$. Therefore, a completely secret permutation should be found after roughly e trials.

* This work was performed while visiting NEC Research Institute in the spring of 1992.

Now consider the scenario where the members of this family cannot be gathered in a room to do the "names-in-the-hat" technique, for instance if some of them live abroad in several different countries. How can such a permutation be chosen locally, for instance by a single person without that person knowing the chosen permutation but knowing that it has no fixed point?

1.1 Related Work

Cryptography and card playing have a long history of connections. There has been substantial work on implementing card games using cryptography [SRA81, GM82, BF83, FM85, Yun85, Cré86, Cré87, GMW87, CCD88, BOGW88, RB]. Conversely, a number of researchers have considered mechanisms for implementing cryptographic primitives based on card games. Winkler [Win81a, Win81b, Win83] shows how two bridge players can securely communicate during the bidding process. More recently, Fischer, Paterson and Rackoff [FPR91] and Fischer and Wright [FW92, FW93] give a number of secret-key exchange protocols based on random card deals. Den Boer [den90] gives a protocol by which two parties may securely compute the AND function, based on the ability to make an *oblivious cut* on a deck of cards. We also base our protocol on oblivious cuts, and use a modified form of the secure AND protocol as a subroutine.

The novel contribution of our work is to provide a new scenario of a single person using cryptographic techniques as building blocs for playing sophisticated solitary games.

1.2 The Scenario

We consider this question in an *honest but non-oblivious* scenario. One person is going to be responsible for picking this permutation and will do this following a protocol we describe (it does not make much sense to try to prevent someone to cheat himself). At no point this person will be asked to forget information it has seen or can deduce from what it saw. Nevertheless we assume that an operation such as choosing a secret random cyclic shift of a set of objects (you may think of randomly cutting a deck of cards) is available to that person in order to create (unknown) randomness in his data. We will show how this person can pick a random permutation on the numbers $1, 2, ..., n$ and verify that it has no fixed point, learning no information whatsoever about which permutation was chosen. We qualify this process of "discreet". We also generalize this problem to the extent that we show how any "solitary game" can be played "discreetly" as long as there exist some polynomial size circuit to describe it.

We work with the following alphabet (each value can be thought as a suit in a deck of cards):

$$\left\{\boxed{\clubsuit}, \boxed{\heartsuit}, \boxed{\diamondsuit}, \boxed{\spadesuit}, \boxed{?}\right\}$$

The value $\boxed{?}$ representing any of the first four but face down on the table. We assume that all copies of one of these 5 elements are indistinguishable from one another. We define two notations on the basic elements for the rest of this paper:

Definition 1 $(c_1c_2...c_k)$**.** For any symbols $c_1, c_2, ...c_k$, we write $(c_1c_2...c_k)$ to represent the elements of $\{c_1c_2...c_k, c_2c_3...c_kc_1, .., c_kc_1...c_{k-1}\}$, that is any cyclic permutation of $c_1c_2...c_k$.

indeed $(c_1c_2...c_k)$ is the equivalence class of strings equivalent up to a cyclic permutation.

Definition 2 $\langle c_1c_2...c_k\rangle$**.** For any symbols $c_1, c_2, ...c_k$, we write $\langle c_1c_2...c_k\rangle$ to express the fact that this string is obtained by a random cyclic shift, meaning that it is replaced by a random element of $(c_1c_2...c_k)$.

for instance,

$$\left\langle \boxed{\heartsuit}\boxed{\clubsuit}\boxed{\diamondsuit}\boxed{\spadesuit} \right\rangle \to \boxed{\diamondsuit}\boxed{\spadesuit}\boxed{\heartsuit}\boxed{\clubsuit}.$$

2 A Solution to the "no-fixed Point" Problem

For the solution to our first problem we use the following trivial coding:

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

A sequence of n bits is associated to each participant p_i. Initially, to p_i associate the sequence of n $\boxed{\clubsuit}$ except in position i where it is a $\boxed{\heartsuit}$.

$$\boxed{\begin{array}{rl}
p_1: & \boxed{\heartsuit}\boxed{\clubsuit}\cdots\boxed{\clubsuit}\cdots\boxed{\clubsuit}\boxed{\clubsuit} \\
p_2: & \boxed{\clubsuit}\boxed{\heartsuit}\cdots\boxed{\clubsuit}\cdots\boxed{\clubsuit}\boxed{\clubsuit} \\
 & \vdots \\
p_i: & \boxed{\clubsuit}\boxed{\clubsuit}\cdots\underbrace{\boxed{\heartsuit}}_{i}\cdots\boxed{\clubsuit}\boxed{\clubsuit} \\
 & \vdots \\
p_n: & \boxed{\clubsuit}\boxed{\clubsuit}\cdots\boxed{\clubsuit}\cdots\boxed{\clubsuit}\boxed{\heartsuit}
\end{array}}$$

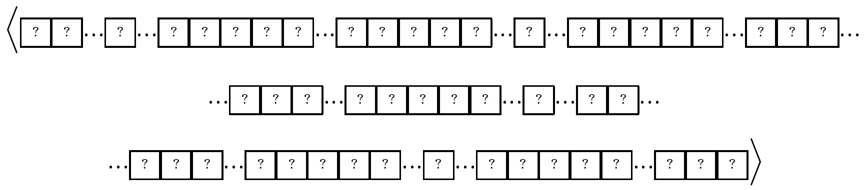
Then construct a long sequence by putting each of these sequences side by side, separated by markers made of $\frac{n}{2}$ $\boxed{\diamondsuit}$'s followed by $\frac{n}{2}$ $\boxed{\spadesuit}$'s

$$\boxed{\heartsuit}\boxed{\clubsuit}\cdots\boxed{\clubsuit}\cdots\boxed{\clubsuit}\boxed{\clubsuit}\overbrace{\boxed{\diamondsuit}\boxed{\diamondsuit}\boxed{\diamondsuit}\cdots\boxed{\spadesuit}\boxed{\spadesuit}\boxed{\spadesuit}}^{marker}\boxed{\clubsuit}\boxed{\heartsuit}\cdots\boxed{\clubsuit}\cdots\boxed{\clubsuit}\boxed{\clubsuit}\overbrace{\boxed{\diamondsuit}\boxed{\diamondsuit}\boxed{\diamondsuit}\cdots\boxed{\spadesuit}\boxed{\spadesuit}\boxed{\spadesuit}}^{marker}\cdots$$

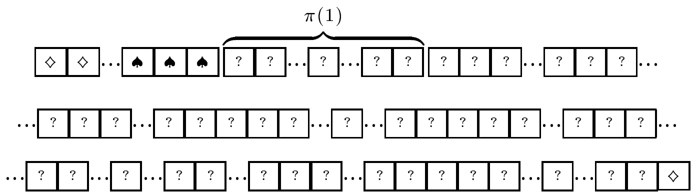
$$\cdots\underbrace{\boxed{\diamondsuit}\boxed{\diamondsuit}\boxed{\diamondsuit}\cdots\boxed{\spadesuit}\boxed{\spadesuit}\boxed{\spadesuit}}_{marker}\boxed{\clubsuit}\boxed{\clubsuit}\cdots\underbrace{\boxed{\heartsuit}}_{i}\cdots\boxed{\clubsuit}\boxed{\clubsuit}\cdots$$

$$\cdots\underbrace{\boxed{\diamondsuit}\boxed{\diamondsuit}\boxed{\diamondsuit}\cdots\boxed{\spadesuit}\boxed{\spadesuit}\boxed{\spadesuit}}_{marker}\boxed{\clubsuit}\boxed{\clubsuit}\cdots\boxed{\clubsuit}\cdots\boxed{\clubsuit}\boxed{\heartsuit}\underbrace{\boxed{\diamondsuit}\boxed{\diamondsuit}\boxed{\diamondsuit}\cdots\boxed{\spadesuit}\boxed{\spadesuit}\boxed{\spadesuit}}_{marker}$$

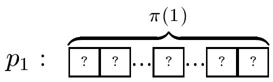
Apply them a random cyclic shift



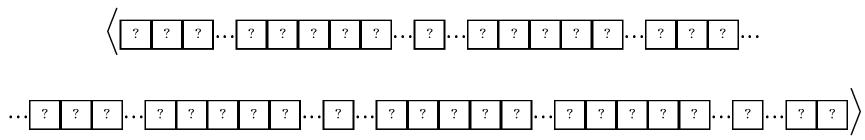
If the first element of the result is a $\boxed{\heartsuit}$ or a $\boxed{\clubsuit}$ then hide it and generate another random cyclic shift until you find a $\boxed{\diamondsuit}$ or $\boxed{\spadesuit}$. When the first element is a $\boxed{\diamondsuit}$ then open values forward until you show $\frac{n}{2}$ $\boxed{\spadesuit}$'s and then enough values backwards to show $\frac{n}{2}$ $\boxed{\diamondsuit}$'s. When you find a $\boxed{\spadesuit}$ first, proceed in the reverse order.



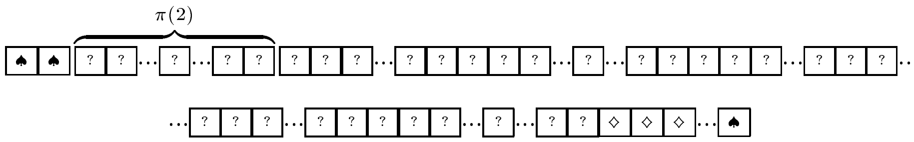
Get rid of the marker and extract the random entry for $\pi(1)$ located in the n values following the opened marker and associate it to p_1.



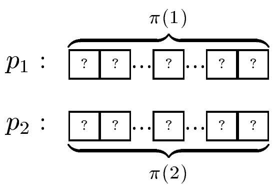
Apply the same process to the remaining values



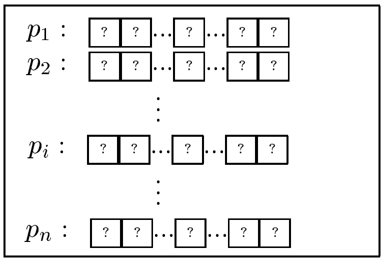
in order to select a random entry for $\pi(2)$.



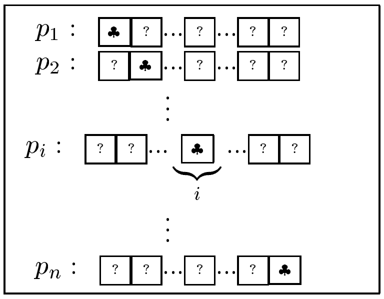
Associate this random entry to p_2



Repeat this process $n-2$ more times an obtain values for $\pi(3), ..., \pi(n)$ and associate each $\pi(i)$ to each p_i.



and check that the permutation π generated has no fixed point by opening the diagonal of this table and checking that it contains only $\boxed{\clubsuit}$'s



After doing so, you can put each sequence of values in an envelope and mail them out to the participants, telling them the correspondence between the n possible sequences and the people. You know for sure that nobody will get its own name and you know nothing at all about the permutation except for that fact.

3 A More Elaborate Problem: No Short Cycles

We now generalize the "no-fixed point" problem in non-trivial ways that will lead us to developing a general theory about what can be done discreetly by oneself.

Suppose that in order to make the exchange more diversified we disallow short cycles of length at most k in the permutation, for some constant k. This constraint makes the problem much more complicated. We first deal, in an ad hoc fashion, with the case $k = 2$ and build tools useful in the general scenario with $k > 2$.

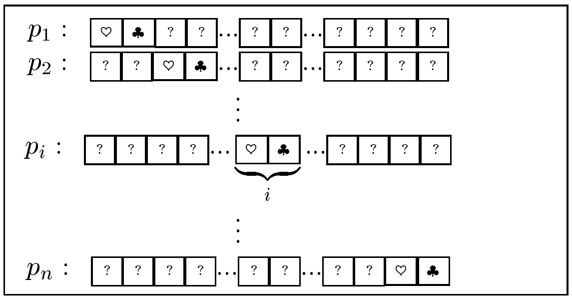
It is very easy to see that for any $k < n$ the probability that a random permutation will have no cycle of length at most k is at least $\frac{1}{n}$. This is because the number of permutations with a single cycle of length n is $(n-1)!$, while the total number of permutations is $n!$. Therefore we can generate permutations with no cycles up to length k simply by picking a random permutation and checking it. On average, after at most n trials, one will work.

3.1 Detecting Two-Cycles

The basic observation is that we would like, for each pair i, j, to check whether $\pi(i) = j$ and $\pi(j) = i$ without learning these values, of course. Basically, what we need is to be able to perform the logical "AND" of positions (i, j) and (j, i) from the table above without learning them. This is possible using den Boer's "Match Making" trick [den90]. To use this we must change our coding to

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

This does not change much about what we did so far, except that we use twice as many values to do the same job. Wherever, $\boxed{\clubsuit}$'s and $\boxed{\heartsuit}$'s were used in the past to represent 0's and 1's, we use our new coding instead (we also double the size of the markers). For instance to check for fixed points now involves opening two values per entry



DenBoer's trick is used as follows to compute the logical "AND" of two secret bits coded as above.

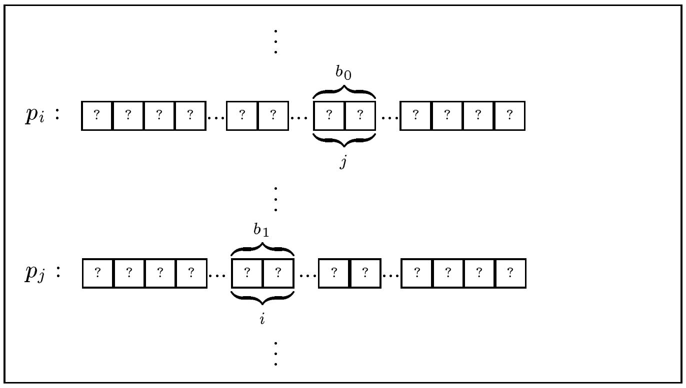
Let b_0 and b_1 be two secret bits for which we would like to find out their logical "AND". Put b_0, b_1 and a $\boxed{\clubsuit}$ side by side

$$\underbrace{\boxed{?}\boxed{?}}_{b_0}\ \underbrace{\boxed{?}\boxed{?}}_{b_1}\ \boxed{\clubsuit}$$

After hiding the $\boxed{\clubsuit}$, swap b_1's values and randomly shift the 5 values cyclically.

$$\Big\langle \boxed{?}\boxed{?}\ \overbrace{\boxed{?}\boxed{?}}^{\rightleftharpoons}\ \boxed{?} \Big\rangle$$

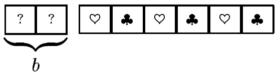
If the resulting sequence has its two $\boxed{\heartsuit}$'s side by side, $(\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit})$ it means that $b_0 = b_1 = 1$ and otherwise $(\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\heartsuit}\boxed{\clubsuit})$ it means that at least one of them was a 0. We can use this trick in order to check whether $\pi(i) = j$ and $\pi(j) = i$



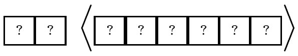
Unfortunately, doing so for a single pair i, j will destroy the data in a non-recoverable way. Therefore we need a mechanism to duplicate a bit in order to compare copies of bits and save some copy for further use.

3.2 Copying a Bit

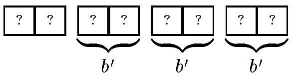
Here is how one can make secret copies of a bit: Starting from a bit b at the left, put an alternation of 6 $\boxed{\heartsuit}$'s and $\boxed{\clubsuit}$'s to its right.



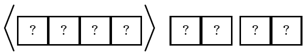
After hiding the 6 rightmost values, apply a random cyclic shift to them.



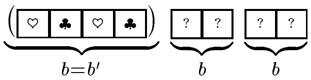
Because of the alternation that was put there in the beginning you know that each of the 3 pairs on the right represent a same bit b', but no longer know the value of b' because of the random cyclic shift.



Now randomly shift the 4 leftmost values.



Open the 4 leftmost values; if the sequence you see is alternating then it means that $b = b'$ and therefore the 4 rightmost values form 2 copies of b.



Otherwise, when the sequence you see is not alternating it means that $b \neq b'$ and therefore the 4 rightmost values form 2 copies of $\bar{b}$.

$$\underbrace{\left(\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\right)}_{b \neq b'}\ \underbrace{\boxed{?}\boxed{?}}_{\bar{b}}\ \underbrace{\boxed{?}\boxed{?}}_{\bar{b}}$$

In order to get copies of b, simply swap the values in each of the 2 rightmost pairs.

$$\left(\boxed{\heartsuit}\boxed{\heartsuit}\boxed{\clubsuit}\boxed{\clubsuit}\right)\ \underbrace{\overbrace{\boxed{?}\boxed{?}}^{\rightleftharpoons}}_{b}\ \underbrace{\overbrace{\boxed{?}\boxed{?}}^{\rightleftharpoons}}_{b}$$

3.3 Detecting k-Cycles

In order to detect cycles longer than 2, it is necessary to perform more complicated computations on the secret bits. Indeed, it is nice to be able to evaluate "AND" gates but, as it is, we cannot even use the result of such a gate in a further secret computation because the answer is not in the same format as the data (a bit represented by a pair of values). Therefore we now show another tool to compute a pair of values (discreetly) that will satisfy some relation with two original pair of values (for instance the later represents the "AND" of the former). Equipped with such a tool, we can easily check for k-cycles of any length simply by designing a circuit that checks the length of all the cycles is at least k. (this is at most an n^3 algorithm)

3.4 Evaluating Logical Gates

Starting with two secret bits b_0, b_1, we show how to create a new secret bit for $b_0 \wedge b_1$. First, by the result of section 3.2 we can easily make copies of b_0, b_1 which we will use later in order to preserve the originals. Call x_0, y_0, x_1, y_1 the values of copies of b_0 and b_1 as follows:

$$\overbrace{\underbrace{\boxed{?}}_{x_0}\ \underbrace{\boxed{?}}_{y_0}}^{b_0}\ \overbrace{\underbrace{\boxed{?}}_{x_1}\ \underbrace{\boxed{?}}_{y_1}}^{b_1}$$

Then, generate a set of 4 closed values as an alternation of $\boxed{\diamondsuit}$ and $\boxed{\spadesuit}$:

$$\boxed{\diamondsuit}\boxed{\spadesuit}\boxed{\diamondsuit}\boxed{\spadesuit}$$

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

$$\underbrace{\boxed{?}}_{c_0}\ \underbrace{\boxed{?}}_{d_0}\ \underbrace{\boxed{?}}_{c_1}\ \underbrace{\boxed{?}}_{d_1}$$

and build two decks as follows:

$$\underbrace{\boxed{?}}_{x_0}\ \underbrace{\boxed{?}}_{c_0}\ \underbrace{\boxed{?}}_{y_0}\ \underbrace{\boxed{?}}_{d_0}\ \text{ and }\ \underbrace{\boxed{?}}_{c_1}\ \underbrace{\boxed{?}}_{x_1}\ \underbrace{\boxed{?}}_{y_1}\ \underbrace{\boxed{?}}_{d_1}\ \boxed{\heartsuit}\boxed{\clubsuit}.$$

After random cyclic shift of the first four

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

two possibilities may occur when opened:

$$\left(\boxed{\heartsuit}\boxed{\diamondsuit}\boxed{\clubsuit}\boxed{\spadesuit}\right)\ \text{ or }\ \left(\boxed{\heartsuit}\boxed{\spadesuit}\boxed{\clubsuit}\boxed{\diamondsuit}\right).$$

Repeat random cyclic shifts of the second deck

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

until the first value on top is the same as the value following the $\boxed{\heartsuit}$ in the first deck:

$$\boxed{\diamondsuit}\ \underbrace{\boxed{?}\boxed{?}}_{b_0 \wedge b_1}\boxed{?}\boxed{?}\boxed{?}$$

in the first case or

$$\boxed{\spadesuit}\ \underbrace{\boxed{?}\boxed{?}}_{b_0 \wedge b_1}\boxed{?}\boxed{?}\boxed{?}$$

in the second case. The next two values after that will contain $b_0 \wedge b_1$.

Similar techniques will also work to build other gates. To do "OR" gates replace

$$\underbrace{\boxed{?}}_{c_1}\ \underbrace{\boxed{?}}_{x_1}\ \underbrace{\boxed{?}}_{y_1}\ \underbrace{\boxed{?}}_{d_1}\ \boxed{\heartsuit}\boxed{\clubsuit}$$

by

$$\underbrace{\boxed{?}}_{d_1}\ \underbrace{\boxed{?}}_{x_1}\ \underbrace{\boxed{?}}_{y_1}\ \underbrace{\boxed{?}}_{c_1}\ \boxed{\clubsuit}\boxed{\heartsuit}$$

and for "XOR" gates by

$$\underbrace{\boxed{?}}_{d_1}\ \underbrace{\boxed{?}}_{x_1}\ \underbrace{\boxed{?}}_{y_1}\ \underbrace{\boxed{?}}_{c_1}\ \underbrace{\boxed{?}}_{y_1}\ \underbrace{\boxed{?}}_{x_1}.$$

These operations combined with negation (flipping the values representing a bit) will suffice to simulate discreetly our circuit to check k-cycles.

3.5 Property Verifiable with Poly-size Circuits

It is clear that any similar property that can be described as an easy to evaluate circuit can be applied to a random permutation. Our techniques therefore allow us to generate discreetly permutations satisfying a certain condition $\mathcal{C}$ as long as

- there is a non-negligible probability of picking a random permutation satisfying $\mathcal{C}$
- $\mathcal{C}$ can be described as an easy to evaluate boolean circuit.

4 General Results

The tools that we have developed in section 3 are indeed very powerful. We now explain some of the things we can do with them.

4.1 Generating Permutations

Let's first focus on our earlier problem of generating permutations with a specific property.

4.2 Property Constructible with Poly-size Circuits

An extension of the result of the previous section is that it is possible to generate permutations satisfying certain properties as long as we can find a probabilistic boolean circuit that will output such a permutation with the correct distribution. What we suggested in the previous section is a particular case of this technique. In order to accomplish this we need one last simple tool: secret random bits. (Generating a random bit is simple: apply a random cyclic shift to a pair of hidden $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$.)

4.3 Playing Games

Finally, we observe that any games such as POKER, BRIDGE and other card games can be played in solitary by describing the strategies of one's opponents as (probabilistic) boolean circuits. The strategies of the opponents can therefore be applied discreetly and played against as if playing with real opponents without learning extra information.

Of course, we do not expect anybody to really implement this idea with cards since it would take a tremendous amount of cards... and time...

5 Remarks and Open Questions

- A two-symbol alphabet suffices to achieve our result.(with a more complex construction)
- We have considered more general primitives such as *shuffling* but conjecture that the same result is impossible in that model.
- Another question is about the minimal number of values that must be randomly shifted through our protocols. We believe 4 suffice (even with a two-symbol alphabet).
- Several of the techniques used in section 3 and 4 are Las Vegas: it may take many iterations before the proper condition is met. This can be avoided at the price of using much longer sequences. (more details available in the final paper) One last open question is to achieve the same result keeping everything efficient.

Acknowledgments

Claude would like to thank Claire Kenyon, whose family-in-law, spread around the world, inspired this research. We would like to thank Charles H. Bennett for his highest enthusiasm in the early days of our result.

References

[BF83] I. Bárány and Z. Füredi. Mental poker with three or more players. *Information and Control*, 59:84–93, 1983.

[BOGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for fault-tolerant distributed computing. In *Proc. 20th ACM Symposium on Theory of Computing*, pages 1–10, Chicago, 1988. ACM.

[CCD88] D. Chaum, C. Crépeau, and I. Damgård. Multi-party unconditionally secure protocols. In *Proc. 20th ACM Symposium on Theory of Computing*, Chicago, 1988. ACM.

[Cré86] C. Crépeau. A secure poker protocol that minimizes the effects of player coalitions. In H. C. Williams, editor, *Advances in Cryptology: Proceedings of Crypto '85*, volume 218 of *Lecture Notes in Computer Science*, pages 73–86. Springer-Verlag, 1986.

[Cré87] C. Crépeau. A zero-knowledge poker protocol that achieves confidentiality of the players' strategy or how to achieve an electronic poker face. In A. M. Odlyzko, editor, *Advances in Cryptology: Proceedings of Crypto '86*, volume 263 of *Lecture Notes in Computer Science*, pages 239–247. Springer-Verlag, 1987.

[den90] B. denBoer. More efficient match-making and satisfiability: The five card trick. In *Advances in Cryptology: Proceedings of Eurocrypt '89*, volume 434 of *Lecture Notes in Computer Science*, pages 208–217. Springer-Verlag, 1990.

[FM85] S. Fortune and M. Merrit. Poker protocols. In G. R. Blakley and D. C. Chaum, editors, *Advances in Cryptology: Proceedings of Crypto '84*, volume 196 of *Lecture Notes in Computer Science*, pages 454–464. Springer-Verlag, 1985.

[FPR91] M. J. Fischer, M. S. Paterson, and C. Rackoff. Secret bit transmission using a random deal of cards. In *Distributed Computing and Cryptography*, pages 173–181. American Mathematical Society, 1991.

[FW92] M. J. Fischer and R. N. Wright. Multiparty secret exchange using a random deal of cards. In *Advances in Cryptology: Proceedings of Crypto '91*, volume 576 of *Lecture Notes in Computer Science*, pages 141–155. Springer-Verlag, 1992.

[FW93] M. J. Fischer and R. N. Wright. An Efficient Protocol for Unconditionally Secure Secret Key Exchange, In *Proc. 4th Annual Symposium on Discrete Algorithms*, January, 1993, 475–483.

[GM82] S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Proc.* 14*th ACM Symposium on Theory of Computing*, pages 365–377, San Francisco, 1982. ACM.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game, or: A completeness theorem for protocols with honest majority. In *Proc.* 19*th ACM Symposium on Theory of Computing*, pages 218–229, New York City, 1987. ACM.

[RB] T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority, In *Proc.,* 21*st ACM Symposium on Theory of Computing*, 1989.

[SRA81] A. Shamir, R. L. Rivest, and L. M. Adleman. Mental poker. In D. Klarner, editor, *The Mathematical Gardner*, pages 37–43. Wadsworth, Belmont, California, 1981.

[Yun85] M. Yung. Cryptoprotocols: Subscription to a public key, the secret blocking and the multi-player mental poker game. In G. R. Blakley and D. C. Chaum, editors, *Advances in Cryptology: Proceedings of Crypto '84*, volume 196 of *Lecture Notes in Computer Science*, pages 439–453. Springer-Verlag, 1985.

[Win81a] P. Winkler. Cryptologic techniques in bidding and defense: Parts I, II, III and IV. In *Bridge Magazine*, April–July 1981.

[Win81b] P. Winkler. My night at the Cryppie club. In *Bridge Magazine*, 60–63, August 1981.

[Win83] P. Winkler. The advent of cryptology in the game of bridge. In *Cryptologia*, 7(4):327–332, October 1983.