A Secure Poker Protocol
that Minimizes the Effect of Player Coalitions

Claude Crépeau

Département d'informatique et de recherche opérationnelle
Université de Montréal
C.P. 6128 succursale "A", Montréal
Québec, Canada, H3C 3J7

1. Introduction

What can we expect from a poker protocol? How close to reality can we come?

From the outset of this research, we realized that a cryptographic protocol could achieve more security than its real life counterpart (with physical cards). But every protocol proposed until now was far from offering all the possibilities of a real deck of cards or could not acheive the full security we were expecting.

Since Shamir, Rivest and Adleman first stated a solution to the mental poker problem [SRA], many protocols trying to implement a fully secure game have been proposed. Although SRA proved in the two player case that such a solution is not possible from an information theoretic point of view, such solutions might be possible when the players' computational power is limited. The leakage of partial information, found by Lipton [Li], in the initial SRA protocol was fixed by Goldwasser & Micali [GM1], in the two player case, using probabilistic encryption. Unfortunately this scheme did not extend to a larger number of players. No complete solution to the multi-player version of the problem is yet known. All proposals make special assumptions, like the players' inability to establish secret communications [Yu]&[BF] or the existence of a trusted third party [FM].

To conceive a complete poker protocol, some constraints must be followed :

- Uniqueness of cards
- Uniform random distribution of cards
- Absence of trusted third party
- Cheating detection with a very high probability
- Complete confidentiality of cards
- Minimal effect of coalitions
- Complete confidentiality of strategy

Each card must appear once and only once, either in the deck or in the hand of one player. The only case when a card may appear more than once must be the result of some detectable cheating.

The hand of each player must depend on decisions made by every players, so that none of them has any control on his hand or on his opponents'. Every possible hand must have an equal probability and be accessible to all players.

No trusted party may be assumed, since any human can be bribed, and no machinery is entirely safe because no tamper proof device can be achieved.

Any attempt to cheat must be detected. The probability that a player may cheat without being detected must decrease very fast (exponentially) with respect to some security parameter that the players must decide before the game. Also the amount of work to accomplish the protocol should increase reasonably (polynomially) with respect to this parameter. The value of this parameter will depend on the confidence the players have towards each other, and the maximum computation power they can achieve.

No partial or total information about any card from the deck may be obtainable without the aproval of every opponent. Also, no information may be obtained from a player's hand without his approval.

When more than two players are involved, some players could establish secret communication and exchange all their knowledge about the game, the protocol or any secret data involved in the protocol. Notheless they should not be able to take advantage of this. This information should be equivalent to the cards they separatly have in their hands. In other words, as long as one player is not cheating, nobody can learn more about his hand, or about the cards in the deck, than what they can deduce from the cards in their coalition.

Finally, it is strategicly very important in the game of poker that the loosing players may keep their cards secret at the end of a hand. All the concept of bluffing is based on this fact. Therefore, an ideal protocol should neither force the players to reveal their hands nor *any* information leading to some knowledge about them.

Our protocol fully implements a secure game achieving the first six properties. Only the confidentiality of the strategy remains unsolved, as all players will be requested to reveal all their information in the detection of cheating phase of this protocol.

This protocol is based on the implementation of the new concept of Hiding-Revealing transactions. These transactions enable a party A to pick a value from a set known to a party B without letting B know which element A has picked.

Our multi-player protocol is a direct extension of our two player version. Let us first describe this simplified protocol. Suppose that P_1 and P_2 want to play a fair game of poker. Assume that each of them will cheat if he can do so (without being detected).

Assume a correspondance between the cards and the set $\{1,2,3,...,51,52\}$. So from now on, numbers from 1 to 52 will be used to describe the cards themselves.

2. Cards shuffling

To shuffle the deck of cards P_1 picks at random a permutation π_1 of $\{1,2,3,...,51,52\}$ and P_2 picks π_2. Accessing any card will be done via these two permutations. Each player P_i locks his permutation π_i using a locking function that must satisfy some special properties described below (which two well known cryptosystems will be shown to satisfy). Then P_i posts the locked version of π_i. A card will be accessed by $\pi_2(\pi_1(k))$ for $k \in \{1,2,3,...,51,52\}$. Since a player doesn't know the permutation of his opponent, the value $\pi_2(\pi_1(k))$ cannot be predicted by either player. To get a card, a player would like to compute $\pi_2(\pi_1(k))$ from k and the encoded permutations (with the collaboration of his opponent) without revealing the outcome of his calculation. This is clearly easy for P_2, but tricky for P_1.

3. Requirements

To play the game, each player will have to pick four functions, (L,U,H,R) according to the rules described below. Briefly, these functions will be used in the protocol to transfer information from one player to another. Suppose a player owns a set of values V. He will show a perfectly hiden version of his elements using the locking function L. The opponent will be able to select a value from V, without revealing which one, by choosing a locked element c, hiding it with H, asking the first player to use U in order to release some trap-door information about $H(c)$, and finally inverting the hiding he did previously, using R on $U(H(c))$ to read the value he has chosen. If H and R have some "good" properties, then the owner of the set V will not be able to determine the value chosen by his opponent, nor will his opponent learn anything on the other values in V.

The following properties summarize sufficient conditions to built our protocol. This general approach is followed by two fundamentally different implementations. The presentation is done in a general framework in order to avoid mixing practical implementation details with the high level aspects of the scheme.

Let V be the value space (in this case $\{1,2,3,...,51,52\} \subseteq V$).
Let S be the seed space.
Let C be the coded (locked) message space.
Let K be the unlocking key space.
Let M be the mask space.
Let A be the ambiguous value space.

Define

$L:V\times S\rightarrow C$	(locking function)
$U:A\rightarrow K$	(unlocking function)
$H:M\times C\rightarrow A$	(hiding function)
$R:M\times C\times K\rightarrow V$	(revealing function)

such that

(the full meaning of the following properties will become clear in the next sections)

1) $\forall v_1,v_2\in V, \forall s_1,s_2\in S,\ \ L(v_1,s_1)=L(v_2,s_2) \implies v_1=v_2$

2) $\forall v\in V, \forall_{\mathbf{P}}\, s\in S$, it is computationally infeasible to compute any information about v from $L(v,s)$.

3) $\forall v\in V, \forall_{\mathbf{P}}\, s\in S$, it is computationally infeasible to compute s from v and $L(v,s)$.

4) $\forall m\in M, \forall v\in V, \forall s\in S,\ \ R(m,L(v,s),U(H(m,L(v,s))))=v$

5) $\forall a\in A, \forall v_1,v_2\in V, \forall s_1,s_2\in S,\ \ \#\{m\in M \mid H(m,L(v_1,s_1))=a\} = \#\{m\in M \mid H(m,L(v_2,s_2))=a\}.$

6) $\forall v \in V, \forall_{\mathbf{P}} (m,s) \in M \times S$, it is computationally infeasible to compute m given $H(m, L(v,s))$ and $L(v,s)$.

7) Their exists some polynomial time algorithm for each of L, U, H, R, but U includes trap-door information which is computationally infeasible to derive from L, H, R.

Here $\forall_{\mathbf{P}} x \in X$ means:
"for all $x \in X$, except a number of elements smaller than any polynomial fraction in $\log(\#X)$".

4. Protocol

Assuming the existence of such functions (two implementations are given later), we describe our protocol. Let $i \in \{1,2\}$.

Preparation Protocol

> Each player P_i :
> Step 1. Defines $V_i, S_i, C_i, K_i, M_i, A_i$.
> Step 2. Defines L_i, U_i, H_i, R_i.
> Step 3. Picks π_i a permutation of $\{1,2,...,52\}$.
> Step 4. Posts L_i, H_i, R_i.
> Step 5. Picks $s_{i,1}, s_{i,2}, \ldots, s_{i,52} \in S_i$ at random.
> Step 6. Posts $l_i(v) = L_i(\pi_i(v), s_{i,v})$, $1 \le v \le 52$.

From property 2, revealing $L_i(\pi_i(v), s_{i,v}), 1 \le v \le 52$, leaks no information on $\pi_i(v)$. But since only 52 values are possible for $\pi_i(v)$, it would be easy to determine $\pi_i(v)$ from $s_{i,v}$ if the latter were easily computable, by comparing $L_i(i, s_{i,v}), L_i(2, s_{i,v}), ..., L_i(52, s_{i,v})$ with $l_i(v)$. This is why property 3 is essential.

Initially, every $v \in \{1,2,3,...,52\}$ is marked as "free". Suppose P_2 wants to get a card,

Card Reading Protocol 1

> Step 1. P_2 picks at random a "free" value $v \in \{1,2,3,...,52\}$ and marks v as "used".
> Step 2. P_2 asks P_1 for the value of $\pi_1(v)$.
> Step 3. P_1 reveals $\pi_1(v)$ to P_2.
> Step 4. P_2 secretly computes $\pi_2(\pi_1(v))$, which is his card.

At the end of the game, P_1 will be able to produce a proof that the value revealed ($\pi_1(v)$) was correct by showing $s_{1,v}$ so that P_2 can compare $L_1(\pi_1(v), s_{1,v})$ with $l_1(v)$. P_1 cannot cheat on this, since $l_1(v)$ uniquely determines $\pi_1(v)$ by property 1.

At the end of the game, if P_2 claims to have the card $\pi_2(\pi_1(v))$ in his hand, he will be able to produce a proof of this by revealing $s_{2,\pi_1(v)}$ so that P_1 can then compare $L_2(\pi_2(\pi_1(v)), s_{2,\pi_1(v)})$ with $l_2(\pi_1(v))$. Again, P_2 cannot cheat on this, due to property 1. So this transaction is fully secured.

Now suppose P_1 wants to get a card,

Card Reading Protocol 2a (not sufficient in general)

> Step 1. P_1 picks at random a "free" $v \in \{1,2,3,...,52\}$
> and marks it as "used".
> Step 2. P_1 secretly computes $\pi_1(v)$.
> Step 3. P_1 picks $m \in M_2$ and computes $h = H_2(m, l_2(\pi_1(v)))$.
> Step 4. P_1 asks P_2 to compute the key k to h.
> Step 5. P_2 returns $k = U_2(h)$ to P_1.
> Step 6. P_1 computes: $R_2(m, l_2(\pi_1(v)), k) = \pi_2(\pi_1(v))$ { by property 4 }.

Since H has property 5, P_2 gets strictly no information on $\pi_1(v)$ from h since any of the l_2's may have been used with equal probability.

We call this (Step 3 to Step 6) a Hiding-Revealing transaction between P_1 and P_2. In this transaction P_2 has revealed a secret value to P_1, namely $\pi_2(\pi_1(v))$, without knowing which one he has given away. At the end of the game P_1 will be able to check this transaction when asking P_2 to reveal $s_{2,\pi_1(v)}$, P_1 can compare $L_2(\pi_2(\pi_1(v)), s_{2,\pi_1(v)})$ with $l_2(\pi_1(v))$. The only remaining problem is proving to P_2 that P_1 did not fool him in making him decode something else than $H_2(m, l_2(\pi_1(v)))$. How can we force P_1 to respect the protocol?

First, let us see how P_1 could cheat. Suppose P_1 asks P_2 to decode $h' = H_2(m, l_2(\pi_1(v')))$ for $v' \neq v$, instead of h. Then P_1 will get $\pi_2(\pi_1(v'))$ when claiming he has been accessing $\pi_2(\pi_1(v))$. This could be interesting for P_1, for instance, if v' is marked as "used" because $\pi_2(\pi_1(v'))$ is in fact a card in P_2's hand. This would allow P_1 to know one card of his opponent, at cost of not knowing one of his own. But at the end of the game, P_1 will not know what $\pi_2(\pi_1(v))$ is. So if P_1 is asked to reveal all the cards he should have accessed, he won't be able to do so. (In fact P_1 may decide to access the card later in the game but in that case the problem will carry over to this new card he pretends to read).

But P_1 could be more subtle than that. He could try to get partial information on many cards at once. Maybe P_1 claims to follow the protocol, when in fact he is asking P_2 to decode some special value $g = G(l_2(1),...,l_2(52))$ instead of h, hoping that $G'(l_2(1),...,l_2(52), U_2(g))$ returns relevant information (partial or total) on more than one entry of π_2, for some easily computable functions $G: C^{52} \rightarrow A$, $G': C^{52} \times K \rightarrow V^{52}$ he has discovered. This way, he might find out what $\pi_2(\pi_1(v))$ is *and* get additional knowledge on some other cards.

Our general solution to this problem takes advantage of property 6. P_2 will ask P_1, at the end of the game, to reveal $\pi_1(v)$, $s_{1,v}$ and m. This allows P_2 to compare $l_1(v)$ with $L_1(\pi_1(v), s_{1,v})$ and h with $H_2(m, l_2(\pi_1(v)))$. But this is not yet a proof of P_1's fairness. Maybe P_1 computed m *after* he received the answer k in Step 5.

Suppose P_1 uses g as defined earlier. Afetr he gets $k = U_2(g)$ from P_2, maybe he can deduce $\pi_2(\pi_1(v))$, additional information on π_2, *and* some m such that $g = H_2(m, l_2(\pi_1(v)))$. However, if we force P_1 to publish a coded copy of m before making Step 4 of the transaction, then to prove a fair access to $\pi_2(\pi_1(v))$, P_1 would have had to compute m *before* learning k. But this is not possible from property 6 because from g and $l_2(\pi_1(v))$, P_1 cannot compute m. So P_1 is fair if and only if he knew m before the value of k was revealed to him.

To implement this solution, we use a (possibily trap-door), public one-way function O_1 that hides all partial information on its inputs and add one step to our protocol (notice that this modification is not necessary for one of the implementations proposed in section 5) :

Card Reading Protocol 2b

Step 1. P_1 picks at random a "free" $v \in \{1,2,3,...,52\}$
and marks it as "used".
Step 2. P_1 secretly computes $\pi_1(v)$.
Step 3. P_1 picks $m \in M_2$ and computes $h=H_2(m,l_2(\pi_1(v)))$.
Step 4. P_1 asks P_2 to compute the key k to h.
Step 4a. P_1 posts $O_1(m)$.
Step 5. P_2 returns $k=U_2(h)$ to P_1.
Step 6. P_1 computes: $R_2(m,l_2(\pi_1(v)),k) = \pi_2(\pi_1(v))$.

This way, P_2 can check later that P_1 knew m before Step 4 of the transaction. The fact that the one-way function hides all partial information is important. In some implementations it is possible for P_2 to compute $\{m \mid \models i,\ 1 \le i \le 52$ such that $H_2(m,l_2(i))=h\}$. If O_1 did leak some information, then the correct m could be found or the set of possible candidates could be reduced, according to the leaked information, and P_2 would gain some knowledge about $\pi_1(v)$.

During the game, cards can be picked from the deck or discarded just by marking (with "used" or "discarded",respectivly) the appropriate element in $\{1,2,3,...,52\}$. (e.g. to discard $\pi_2(\pi_1(v))$ just mark v as "discarded".)

At the end of the game all secret information must be published as proofs of fairness of the players. But some care must be taken in the implementations where the O_i functions are not used, to avoid revealing some secret item too soon. Otherwise a cheating opponent could forge a fairness proof from what he just learned. So, proofs of fairness must be done in the following way. To prevent a cheater from forging a proof, each player must execute Step 1 before anyone does Step 2 since Step 2 is used to prove that the values revealed in Step 1 were the correct ones. Also each one must execute Step 2 before anyone does Step 3 since learning the s's may be a clue to the successful forging of m.

Proof of Fairness Protocol

Each player P_i reveals:
Step 1. $\pi_1(v),\pi_2(\pi_1(v))$ for each v he has accessed.
Step 2. All $m \in M$ used for some Hiding-Revealing operations.
Step 3. $\pi_i(v)$ and $s_{i,v}$,$1 \le v \le 52$.

This enables the opponent to check the transactions and to be sure that no cheating took place. The generalization to more than two players is found in section 6.

5. Implementations

We now propose two implementations of this protocol. The first is based on RSA [RSA] and the efficient probabilistic encryption scheme of Blum & Goldwasser (BG) [BG]. The second is based on the probabilistic encryption scheme of Goldwasser & Micali (GM) [GM]. This first version matches the general pattern given above.

5.1. Using RSA/BG.

Let P be one of the players. P selects p,q two larges primes; large enough for the least 6 RSA bits to be 1/2+(1/poly(log(pq)))-secure (see [CG]). Let $n=pq$. P selects ε,δ such that $\varepsilon\delta \equiv 1 \ (mod\ \phi(n))$. Then define $V=\{0,1,2,3,...,63\}$, $S=K=M=A=\mathbb{Z}_n^*$, $C=V\times S$. $(x\downarrow n)$ denotes the least n significant bits of x and $\oplus$ denotes the bit-by-bit exclusive-OR.

Functions

$$
\begin{aligned}
&L(v,s)=((s\downarrow 6)\oplus v,(s^{\varepsilon} \bmod n))\\
&U(c)=(s(c)^{\delta} \bmod n)\\
&H(m,c)=(m^{\varepsilon}s(c) \bmod n))\\
&R(m,c,k)=(((m^{-1}k) \bmod n)\downarrow 6)\oplus v(c)\\
&O(m)=BG(m,s)
\end{aligned}
$$

Where m^{-1} is the multiplicative inverse of m $(mod\ n)$ and $c=(v(c),s(c))$. BG is the Blum-Goldwasser encryption function (see [BG]); in this implementation, one can use BG in the following way. P picks a random $s_0\in S$, and compute $s_k=s_{k-1}^{\varepsilon} \bmod n$, $1 \le k \le (|n|/6)$. P then posts $s_{\lfloor\frac{n}{6}\rfloor}$ and $<(s_0\downarrow 6)(s_1\downarrow 6)...(s_{\lfloor\frac{n}{6}\rfloor-1}\downarrow 6)> \oplus m$. Here <...> denotes the concatenation of the given blocks of 6 bits. According to [BG], no partial information about m can efficiently be computed from these two public values. At the end of the game P will have to reveal s, then everybody will be able to compute $s_1,s_2,...$ and recover m by inverting the $\oplus$. It will also be possible to check that this is the correct s_0 since $s_{\lfloor\frac{n}{6}\rfloor}$ is uniquely decodable.

Theorem: L,U,H,R satisfy properties 1 to 7 (assuming inverting RSA is hard).

Proof:

1) $\forall v_1,v_2\in V,\forall s_1,s_2\in S,\ L(v_1,s_1)=L(v_2,s_2)$
$\Rightarrow ((s_1\downarrow 6)\oplus v_1,(s_1^{\varepsilon} \bmod n))=((s_2\downarrow 6)\oplus v_2,(s_2^{\varepsilon} \bmod n))$
$\Rightarrow (s_1^{\varepsilon} \bmod n)=(s_2^{\varepsilon} \bmod n)$ and $(s_1\downarrow 6)\oplus v_1=(s_2\downarrow 6)\oplus v_2$
$\Rightarrow s_1=s_2$ and $(s_1\downarrow 6)\oplus v_1=(s_2\downarrow 6)\oplus v_2$
$\Rightarrow v_1=v_2$

2)Since the 6 least significant bits of s are $1/2+(1/\text{poly}(\log(n)))$-secure.

3)RSA is assumed hard to invert. Being given the last 6 bits of s can at best speed up finding s by a factor of 64.

4)$\forall m\in, \forall v\in V, \forall s\in S$, $H(m,L(v,s)) = ((m^{\varepsilon}s^{\varepsilon}) \bmod n) = ((ms)^{\varepsilon} \bmod n)$
$\Rightarrow U(H(m,L(v,s)))=(ms) \bmod n$
$\Rightarrow R(m,L(v,s),U(H(m,L(v,s)))) =((m^{-1}(ms \bmod n) \bmod n)\downarrow 6) \oplus (s\downarrow 6) \oplus v$
$=((s \bmod n)\downarrow 6) \oplus ((s \bmod n)\downarrow 6) \oplus v = v$

5)$\forall v\in V, \forall s\in S, \forall a\in A$, $\{m\in M \mid H(m,L(v,s))=a\}=\{m\in M \mid (ms)^{\varepsilon} \bmod n=a\}$
$=\{a^{\delta}s^{-1} \bmod n\}$
$\Rightarrow \#\{m\in M \mid H(m,L(v,s))=a\} = 1$

6)Suppose we have $x\in \mathbb{Z}_n^*$. If a polynomial fraction in $\log(\#M\times C)$ values of $H(m,c)$ were easy to invert given c, then we could choose $a_k=s(c_k)r_k^{\varepsilon}x$ for polynomially many random c_k,r_k and with a very high probability, find a solution to $a_{k_0}=H(m,c_{k_0})$ (k_0 is one of the values attempted), using that inversion algorithm. One can check that $mr_{k_0}^{-1}$ would then be a solution to $(mr_{k_0}^{-1})^{\varepsilon}\equiv x \ (mod\ n)$. This would imply we can invert RSA.

7) Clearly L,U,H,R are easy to compute.
□

In this first implementation we need O to solve the cheating problem. But in this next one, no O is required. Data expansion may be greater in this second version but the simplicity obtained worth the difference.

5.2. Using Probabilistic Encryption (GM).

Let P be one of the players. P selects p,q two larges primes. Let $n=pq$. P selects η such that $\left[\frac{\eta}{p}\right]=\left[\frac{\eta}{q}\right]=-1$. (where $\left[\frac{x}{p}\right]$ is the Legendre symbol of x over p). Then define $V=K=\{0,1\}^6 =\{0,1,2,3,...,63\}$, $S=(\mathbb{Z}_n^*)^6$, $C=A=\mathbb{Z}_n^*[+1]^6$ and $M=V\times S$. (where $\mathbb{Z}_n^*[+1]=\{x \mid x\in \mathbb{Z}_n^*$ & $\left[\frac{x}{n}\right]=+1\}$ and $\left[\frac{x}{n}\right]$ is the Jacobi symbol of x over n)

Denote $x=(x_1,x_2,x_3,x_4,x_5,x_6)\in X$,where (x,X) is any of $(v,V),(c,C),(s,S),(k,K)$ or (a,A). Denote also $m=(v(m),s(m))\in M=V\times S$ and $(c \otimes c')_j = (c_j c'_j \bmod n)$.

Functions

$L(v,s)=(\lambda(v_1,s_1),\lambda(v_2,s_2),\lambda(v_3,s_3),\lambda(v_4,s_4),\lambda(v_5,s_5),\lambda(v_6,s_6))$
$U(a)=(\upsilon(a_1),\upsilon(a_2),\upsilon(a_3),\upsilon(a_4),\upsilon(a_5),\upsilon(a_6))$
$H(m,c)=L(m) \otimes c$
$R(m,c,k)=v(m) \oplus k$

Where $\lambda(x,y)=y^2\eta^x \bmod n$ and $\upsilon(x)=\begin{cases} 0 \text{ if } x \textit{ is a quadratic residue mod } n \\ 1 \text{ if } x \textit{ is not a quadratic residue mod } n \end{cases}$. Note that these functions are inspired by those defined in [GM2].

Theorem: L,U,H,R satisfy properties 1 to 7. (assuming the quadratic residuosity assumption (QRA))

Proof:

1) $\forall v,v' \in V, \forall s,s' \in S,\ L(v,s)=L(v',s')$
$\Rightarrow v=v'$ since GM is uniquely decodable.

2) Known property under QRA. (see [GM2])

3) True under QRA. Because the ability to compute s from v and $L(v,s)$ would allow to extract square roots, hence factor n wich is hard under QRA.

4) $\forall v \in V, \forall s \in S, \forall m \in M,\ R(m,L(v,s),U(H(m,L(v,s))))$
$=v(m) \oplus U(H(m,L(v,s)))$
$=v(m) \oplus v(m) \oplus v$ since $\upsilon(s(m)_j^2\eta^{v(m)_j}s_j^2\eta^{v_j})=v(m)_j \oplus v_j,\ 1 \le j \le 6$
$=v$

5) $\forall v \in V, \forall s \in S, \forall a \in A,\ \{m \in M \mid H(m,L(v,s))=a\}=\{m \in M \mid L(m) \otimes L(v,s)=a\}$
$=\{m \in M \mid s(m)_j^2\eta^{v(m)_j} \bmod n = as_j^{-2}\eta^{-v_j} \bmod n,\ 1 \le j \le 6\}$
$=\{m \in M \mid s(m)_j^2\eta^{v(m)_j} \bmod n = x_j\eta^{v'_j} \bmod n,\ 1 \le j \le 6\}$ with $as_j^{-2}\eta^{-v_j}=x_j\eta^{v'_j}$ where x_j is a quadratic residue mod n.
$=\{m \in M \mid s(m)_j^2=x_j \text{ and } v(m)_j=v'_j,\ 1 \le j \le 6\}$
But each of these 6 equations have 4 solutions.
So $\#\{m \in M \mid H(m,L(v,s))=a\}=4^6$.

6) Suppose we have $x \in \mathbb{Z}_n^*[+1]$. If a polynomial fraction in $\log(\#M \times C)$ values of $H(m,c)$ were easy to invert given c, then we could choose $a_{k,1}=c_{k,1}s_{k,1}^2\eta^{v_{k,1}}x \bmod n$, with $a_{k,2},a_{k,3},a_{k,4},a_{k,5},a_{k,6}$ random numbers in $\mathbb{Z}_n^*[+1]$, for polynomially many random c_k,s_k,v_k, and with a very high probability, find a solution to $a_{k_0}=H(m,c_{k_0})$ (k_0 is one of the values attempted), using that inversion algorithm. One can check that $ms_{k_0,1}^{-2}\eta^{-v_{k_0,1}}$ would then be a solution to $(s_{k_0,1}^{-1}s_1(m))^2\eta^{v_1(m)-v_{k_0,1}} \equiv x \pmod n$. This would imply we can decide quadratic residuosity (mod n) (in contradiction with QRA).

7) Clearly L,U,H,R are easy to compute.
$\square$

This implementation is particular because no encoding of $m \in M$ is needed to prove that someone knew m before using it in the Hiding-Revealing transaction. This is because of the next result:

Theorem: m cannot be computed from $U(H(m,c))$ and c, for any $m \in M, c \in C$.

Proof: $U(H(m,c))$ is independent of $s(m)$.

Suppose P_j makes a hiding-revealing transaction with P_i. If he doesn't know m before the transaction he cannot know it after since $s(m)$ cannot be computed from $U(H(m,c))$. So to prove that he knew m before the transaction, P_j just has to prove he knows it at the end of the game, since nothing has revealed this value to him after the transaction or later in the game.

6. Multi-Player protocol

We now extend the previous protocol to the multi-player problem. Suppose that $P_1, P_2, \ldots, P_j$ want to play poker. The preparation protocol is identical to the two-player version. A card will be accessed as $\pi_j(\pi_{j-1}(\pi_{j-2}(\ldots(\pi_2(\pi_1(v)))\ldots)))$ for $v \in \{1,2,3,\ldots, 52\}$. Suppose P_n, $n \in \{1,2,\ldots,j\}$ wants to get a card.

Card reading Protocol 3

Step 1. P_n picks at random $v \in \{1,2,3,\ldots, 52\}$
and marks it as "used".
Step 2. $v_1 = v$
Step 3. FOR $i=1$ TO $n-1$ DO
Step 3.1 P_n asks P_i to reveal $\pi_i(v_i)$;
Step 3.2 P_i answers $\pi_i(v_i)$ publicly;
Step 3.3 P_n sets $v_{i+1} = \pi_i(v_i)$;
Step 4. P_n secretly computes $v_{n+1} = \pi_n(v_n)$;
Step 5. FOR $i=n+1$ TO j DO
Step 5.1 P_n secretly gets $v_{i+1} = \pi_i(v_i)$
using the Hiding-Revealing protocol with P_i;
Step 6. P_n's new card is v_{j+1}

This way, P_n computes $v_{j+1} = \pi_j(\pi_{j-1}(\pi_{j-2}(\ldots(\pi_2(\pi_1(v)))\ldots)))$ and nobody except himself knows $v_{n+1}, \ldots, v_j, v_{j+1}$. All the proofs of fairness described in the two-player version can be used again in the multi-player case. Again depending on the implementation, O_i's may be needed to obtain proofs of fairness in the Hiding-Revealing protocol. Finally the order of revelations at the end of the game must be the same as in the protocol for two players when those O_i's are not used.

7. Security against player coalitions

The main improvement of this protocol is protection against coalitions. If some players form a coalition, they will not get any advantage from it, other than learning each other's hand. Since every card is accessed through each permutation, no subset of the players can know anything about the cards of the other players, other than knowing that they are not the cards within the coalition. Assume some player P_i is not a member of a coalition (by this we mean that P_i does not reveal any private information to any other player), then by the construction of the protocol we know that the values $v_{i+1},...,v_j,v_{j+1}$ are secret and known only to himself for each of his cards. Since the v_{j+1}'s actually identify his cards, nobody has any information on them (unless someone has not followed the protocol but in that case he will be detected at the end of the game). Similarly, no coalition can influence the cards drawn by an honest player.

8. Playing other games

Our new protocol can be extended to play almost any card game, as well as other kind of games, such as Scrabble. Fortune and Merrit [FM] mentioned games that could not be played with their protocol. With our protocol, one can play any game where cards have to be exchanged between players more than once, or where cards may be dealt and discarded many times. Suppose for instance that P_n wants to get a card from P_i's hand.

Card exchange Protocol

> Step 1. P_i reveals, in a random order, a locked version of his cards
> ($L_i(x,s_x)$ for each card x in his hand, with a random seed $s_x \in S_i$).
> Step 2. P_n picks one of them, tells which one to P_i
> Step 3. P_i returns $L_n(x,s)$ with a random seed $s \in S_n$
> where x is the card P_n wants.
> Step 4. P_n recovers x using H_n, U_n, R_n.

This way, only P_i and P_n know which card was exchanged. In fact, everybody will be able to check this operation at the end of the game when P_i reveals the s_x's. Also no information, like the identity of the previous players who ever had the card, is embedded with it. One might think that the following solution is sufficient, but if fact it is not. P_N picks a value that P_i claims as "used" and reads it using the Hinding-Revealing protocol. P_n has to inform P_i of which card he has picked so, that P_i knows that this card is no longer in his hand. But if later P_i have to pick a card from P_n's hand, he will be able to choose the same card P_n had picked before since he knows how to access it.

One can notice a problem with discarding in our standard protocol if the game played allows several dealing and discarding of cards. Suppose cards were dealt and then some were discarded. Suppose we deal some more cards and then discard some of the cards in our new hands (there are variants of poker in which players may ask twice to change cards). On the second discarding operation each player knows if the discarded cards of his opponents, come from the initial hand or from the new one dealt, since the "discard" markers are tagged on to the public values $v \in \{1,2,...,52\}$. This information may be compromising. To solve this, a card should be discarded by revealing a coded version of it and declaring it discarded (This idea was introduced in [Yu]). If P_i wants to discard the card $\pi_j(\pi_{j-1}(\pi_{j-2}(...(\pi_2(\pi_1(v)))...)))$, instead of marking v as "discarded", he does the following:

Discarding Protocol

P_i posts $L_i(v,s)$ for some $s \in S_i$ and declares it "discarded".

At the end of the game, when P_i reveals v and s, the other players will be able to determine which cards P_i has accessed, which are still in his hand and which were discarded during the game. Just like in the exchange of cards, this reveals no information on its origin.

Another interesting feature of this protocol is the ability to return cards into the deck. Initially, each player goes through the Preparation Protocol, exactly as before and uses the other protocols for the other standard operations. Suppose that some players (maybe only one) want to return some cards (maybe only one) into the deck for the n^{th} time. It would not suffice to change their marks from "used" to "free" because this would allow the next person who select one of these cards to know that it had been in someone's hand previously. The entire deck, including the cards just returned to it, must be re-shuffled by:

Card Returning Protocol(part 1)

Each player P_i:
Step 1. Picks $\pi_{i,n}$ a new permutation of $\{1,2,...,52\}$.
Step 2. Picks $s_{i,1,n}, s_{i,2,n}, \ldots, s_{i,104,n} \in S_i$ at random.
Step 3. Posts $l_{i,n}(v) = L_i(\pi_{i,n}(v), s_{i,v,n})$, $1 \le v \le 52$.
Step 4. Posts $l_{i,n}(v+52) = L_i(\pi_{i,n}^{-1}(v), s_{i,v+52,n})$, $1 \le v \le 52$.
Step 5. Sets $\pi_i = \pi_{i,n}$

When this is all done, the players will have to read backwards the new origin of the cards they have under their control (in their hands and among those they have discarded). They will not be able to cheat on this since the other players will check the correspondance between the cards had under each of the $\pi_{i,n}$.

Card returning Protocol(part 2)

Each player P_i, for each card c under his control he wishes to keep,
Step 1. sets $c_j = c$
Step 2. FOR $l = j$ DOWNTO $i+1$
Step 2.1 Reads $\pi_{l,n}^{-1}(c_l)$ using the Hiding-Revealing protocol with P_l ;
Step 2.2 Sets $c_{l-1} = \pi_{l,n}^{-1}(c_l)$;
Step 3. Sets $c_{i-1} = \pi_{i,n}^{-1}(c_i)$;
Step 4. FOR $l = i-1$ DOWNTO 1
Step 4.1 Reads $\pi_{l,n}^{-1}(c_l)$ using the Hiding-Revealing protocol with P_l ;
Step 4.2 Sets $c_{l-1} = \pi_{l,n}^{-1}(c_l)$;
Step 5. Declares c_0 (the origin of c) as "used".

These operations can all be verified when the $\pi_{i,n}$'s and the $s_{i,l,n}$'s, for $1 \leq l \leq 104$, are revealed. Notice that this feature enables the implentation of:"a Scrabble Protocol that minimizes the effect of player coalitions". To do this, change cards into letters and the deck into the box of letters. Then the dealing of letters is similar to the dealing of cards and so on... But since letters can be returned into the box, this last feature is necessary to implement that game.

9. Open Problem

Nothing is quite perfect. There is one thing our protocol cannot do. The strategy of each player is completely revealed at the end of each game since our protocol asks everyone to show every information involved in the protocol. It makes it impossible for the players to bluff. Real poker players would never accept to play such a game. Although we believe such a protocol can be achieved, we do not have a complete solution yet.

10. Acknowledgements

I wish to thank Gilles Brassard, Pierre McKenzie and Jean-Marc Robert for fruitfull discussions and for the numerous comments they made on the protocols.

11. REFERENCES

[BF] Banary, I. and Furedi, Z. "Mental Poker with Three or More Players", in *Information and Control*, 59 (1983), pp. 84-93.

[BG] Blum, M. and Goldwasser, S. "An *Efficient* Probabilistic Public-Key Encryption Scheme which Hides All Partial Information", in *Advances in Cryptology: Proc. of Crypto 84*, G. R. Blakley and D. Chaum, eds., Lecture Notes in Computer Science 196, Springer-Verlag, Berlin, 1985, pp.289-299.

[CG] Chor, B. and Goldreich, O., "RSA/Rabin Least Significant Bits Are 1/2+1/poly(log n) Secure", in *Advances in Cryptology: Proc. of Crypto 84*, G. R. Blakley and D. Chaum, eds., Lecture Notes in Computer Science 196, Springer-Verlag, Berlin, 1985, pp.303-313.

[FM] Fortune, S. and Merrit, M., "Poker Protocols", in *Advances in Cryptology: Proc. of Crypto 84*, G. R. Blakley and D. Chaum, eds., Lecture Notes in Computer Science 196, Springer-Verlag, Berlin, 1985, pp.454-464.

[GM1] Goldwasser, S. and Micali S., "Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information", in *Proceedings of the 14th Annual ACM symp. on Theory of computing*, ACM-SIGACT, May 1982, pp. 365-377.

[GM2] Goldwasser, S. and Micali S., "Probabilistic Encryption", in *J. Comput. System Sci.*, 28 (1984), pp. 270-299.

[RSA] Rivest, R., Shamir, A. and Adleman L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", in *Communications of the ACM* 21,2 (February 1978), pp. 120-126.

[SRA] Shamir, A., Rivest R. and Adleman L., "Mental Poker", MIT Technical Report, 1978.

[Yu] Yung, M., "Cryptoprotocols: Subscription to a Public Key, The Secret Blocking and the Multi-Player Mental Poker Game", in *Advances in Cryptology: Proc. of Crypto 84*, G. R. Blakley and D. Chaum, eds., Lecture Notes in Computer Science 196, Springer-Verlag, Berlin, 1985, pp.439-453.