A Toolbox for Mental Card Games

Christian Schindelhauer

Medizinische Universität Lübeck*

September 17, 1998

Abstract

Mental card games are played without a trusted party and without cards. It is well known that the problem of mental card games can be solved in principle. But the schemes known so far are too messy to be used in practice. Only for the mental poker game a suitable solution is known [Crép 87] that achieves security against player coalition and complete confidentiality of a player's strategy. Here, we present a general-purpose scheme that may be used as basic toolbox for straight-forward implementations of card games.

We present a data structure for cards and decks that is secure against player coalitions and enables standard operations like picking up a card, opening it, and (re-)mixing stacks. Futhermore, we introduce tools for special operations like inserting a card into the deck, splitting the deck, parting the game. The correctness of all operations is testified by zero-knowledge proofs.

Finally, we discuss security problems that are typical for mental card games and suggest solutions to enable all players maximum possible fairness.

1 Introduction

Card games have always been strongly influencing the development of mathematics and computer science. Researchers were interested in unterstanding the structure of randomness, the combinatorics, and the probabilism involved. Nowadays, not only the analysis of the game, but its simulation engages scientific imagination. The simplest example of a mental game is coin flipping by telephone [Blum 81]: Two partners want to generate a random bit, without the assistance of a trusted referee. Both partners are interested in manipulating the outcome of the game.

The problem of how to play mental poker was investigated by [GoMi 82]. Their protocol realizes all necessary features of a poker game. The main drawback is that after the game all cards will be publicized and so the players' strategy will be revealed, which is not acceptable. Another drawback is the exhaustive use of prime numbers: The scheme uses 2×52 pairs of prime numbers and each card is encoded by the quadratic residuosity of 2×6 six numbers w.r.t. two products of prime numbers. Picking up a card implies the publishing of an integer factorization.

Due to lack of space we cannot mention the variety of approaches to mental poker [SRA 78, BaFü 83, FoMe 85, Yung 85, Crép 86, GMW 87] made so far. A sufficient and complete solution of mental poker is presented in [Crép 87]. It achieves: Uniform random distribution of cards, absence of a trusted third party, cheating detection, complete confidentiality of cards, minimal effect of coalition, and complete confidentiality of strategy. The last points include that cards picked up are not published after the game. Furthermore, if k is the number of players, the scheme uses only $2k$ prime numbers as secret keys and k products of two prime numbers as public keys.

Crepeau's poker scheme is based on the *all-or-nothing disclosure of secrets*-protocol (ANDOS) [BCR 87]. Alice knows some t secret strings $a_k \in \{0,1\}^n$. For $\{1,\ldots,t\}$ and $j \in \{1,\ldots n\}$ let

*Institut für Theoretische Informatik, Wallstraße 40, 23560 Lübeck, Germany
email: schindel@informatik.mu-luebeck.de

$a_{k,j}$ denote a single bit. Alice wants to disclose Bob only one of these strings a_k. Bob wants to know this string but without Alice getting any knowledge which string he has chosen. Furthermore, Alice wants to avoid Bob getting more information than included in a_k whilst k remains unknown for Alice.

For this, Bob sends Alice P_σ: This includes a permutation σ over {1, ..., t} and is defined for random $r_{*,*} \in \mathbb{Z}_m^*$ and $a_{*,*} \in \{0,1\}$ as

$$\begin{aligned} P_\sigma := (q_{k,j} \mid \ & k \in \{1,\dots,t\},\ j \in \{1,\dots,n\} \\ & \text{where } q_{k,j} = z_{i,j}(r_{k,j})^2 y^{a_{k,j}} \bmod m \text{ and } i = \sigma^{-1}(k)) \ . \end{aligned}$$

ANDOS [BCR 87]

Step	Alice		Bob
1.	$\boxed{\text{Alice creates public key } m \text{ and } y \in \mathbb{NQR}_m^\circ}$		
2.	Alice chooses nt random numbers $x_{j,k} \in \mathbb{Z}_m^*$ and computes an encoding of her secret strings $b_{*,*}$: $z_{j,k} := (x_{j,k})^2 y^{b_{j,k}} \bmod m$ for $j \in \{1,\dots t\}$, $k \in \{1,\dots n\}$.	$\xrightarrow{z_{*,*}}$	
3.		$\xleftarrow{P_\sigma}$	Bob chooses a permutation σ and constructs P_σ.
4.	$\boxed{\text{Bob proves Alice the correctness } P_\sigma.}$		
5.		$\xleftarrow{k}$	Bob chooses $i \in \{1,\dots,t\}$ and sends $k = \sigma(i)$.
6.	Alice computes for $q_{\mathbf{k},1}, \dots q_{\mathbf{k},n}$ the vector $c_1,\dots,c_n$ such that $c_j := \begin{cases} 1\ , & q_{k,j} \in \mathbb{QR}_m \\ 0\ , & q_{k,j} \notin \mathbb{QR}_m \ . \end{cases}$	$\xrightarrow{c_1,\dots,c_n}$	
7.			Bob computes $b_{i,j} = c_j \oplus a_{k,j}$.

Later on, we will adapt the following protocol verifying that Bob uses a correct formed P_σ.

Proof of P_σ [BCR 87]

Step	Alice		Bob
1.	Alice chooses security parameter s.	$\xrightarrow{s}$ $\xleftarrow{P_1,\ldots,P_s}$	B chooses s permutations $\sigma_1, \ldots, \sigma_s$ over $\{1, \ldots t\}$. Bob produces $P_{\sigma_1}, \ldots, P_{\sigma_s}$.
3.	Alice chooses a random subset $X \subseteq \{1, \ldots s\}$.	$\xrightarrow{X}$	
4.	For all $\ell \in \{1, \ldots s\}$:		
4a.	Alice checks whether σ_ℓ is a correct permutation and whether for all $k \in \{1, \ldots, t\}$ and $j \in \{1, \ldots n\}$ it holds: s $q_{k,j} \equiv z_{i,j}(r_{k,j})^2 y^{a_{k,j}} \pmod{m}$ and $i = \sigma_\ell^{-1}(k)$.	$\xleftarrow{\sigma_\ell, r_{*,*}, a_{*,*}}$	If $\ell \in X$ Bob sends all secret information of P_ℓ and the permutation σ_ℓ.
4b.	Alice checks whether $\sigma_\ell^{-1} \circ \sigma$ is a permutation and whether values $c_{*,*}, e_{*,*}$ are correct.	$\xleftarrow{\sigma_\ell^{-1} \circ \sigma}$ $\xleftarrow{c_{*,*}, e_{*,*}}$	If $\ell \notin X$ Bob sends permutation $\sigma_\ell^{-1} \circ \sigma$. Bob proves that he can can present queries $q_{*,*}$ of P_σ by queries q' in P_ℓ. He succeeds, if he computes in the representation $q_{k,j} \equiv (c_{k,j})^2 y^{e_{k,j}} q'_{\sigma_\ell^{-1}(\sigma(k)),j} \pmod{m}$ $c_{k,j}$ and $e_{k,j}$ and sends them to Alice.

Crepeau uses this ANDOS-scheme and defines the following mental card protocols: Prepare the mental poker game, get a card, detect cheaters, trace restoration, discard, open a card. These protocols must only be used in some sequential ordering, e.g. after drawing a card further mixing is impossible. Discarded cards cannot be reused. Of course this scheme is absolutely sufficient for poker. But it cannot be used for games using more sophisticated basic game steps. Furthermore, some protocols like traces restoration are very expensive with respect to communication and computation.

There is no straightforward way to apply this scheme to general card games, e.g. Crepeau's scheme relies on the uniqueness of cards (which is not given in general). Further, there are many exotic operations that occur in some card games which the mentioned scheme is not able to deal with: Introduction of new cards, drawing a card out of another player's hand, controlling whether a player lays a card of a certain color covered on the stack, etc. It turns out that very simple games like *scrabble* or *old maid* need more complicated protocols than poker.

> [Crép 87]: We have achieved the first complete solution to the mental poker problem. ...In order to solve even more problems of card playing or similar games (such as scrabble), with special operations such as returning cards into the deck, the full power of Boolean circuit simulation suggested in [BrCr 86] can be used. But unfortunately, the resulting protocol is too messy to be explained here.

We modify and extend many ideas and techniques of [Crép 87] and introduce a general toolbox for mental card games that provides efficient and secure communication protocols for transforming every multi-player card game into an efficient and secure mental game. Such a card game is played by k players with contrary interest. The players are separated and communicate via a broadcast channel where each player can send, such that all partners can hear the message and identify its sender. There is no trustable party in the game. Maybe the players have adversary interests, but also some secret coalition of players (exchanging information secretly) may be possible.[1]

[1] Note that in principle we cannot avoid that some players publish their own private cards. We only guarantee that no coalition can find any private card of a player against his will.

We will define a data structure for cards which encodes the type of the card, e.g. 10 of spades. More formally, there is a finite set of types, which can be computed using a function. Whether the types of the cards are unique depends on the rules of the game. A player can only determine the type of a card if all other players allow this. If cards are gathered in stacks, this is simply denoted by tuple of cards $D = (C_1, C_2, \ldots, C_d)$. Only the type of a card is hidden. Every player knows the number and the encoding of all cards.

All players know the rules of the game that may be probabilistic. Every step should be at least checkable by one player who may change during the game. We will discuss how to check all steps according to the rules during the game. For simplicity we do not formalize that aspect. These are some of the possible basic steps that may be used in arbitrary order: Create of new/additional cards, set operation on stacks, split a stack, stack cards, mix a stack, pick up a card, open a card, checking rules of the game that apply to private cards, insert a card at a hidden position of a stack.

This extended abstract is organized as follows: After introducing some notational and number-theoretical background we show the computation of the players' public and private keys. Then we present a masking operation for numbers which forms the cryptographic backbone of the scheme's security. Further, we present the data structure for cards and give protocols for the basic operations on cards and on stacks. In the last section we discuss possibilities of attacking a mental card game and practical issues needed for mental games on computer networks. At last we discuss security issues and present some open problems.

2 Notations and Number-Theoretic Background

Let $\mathbb{Z}_n := \{0, 1, \ldots, n-1\}$ and $\mathbb{Z}_n^* := \{a \in \mathbb{Z}_n \mid \gcd(a,n) = 1\}$, where $\gcd(a,n)$ denotes the greatest common divisor of a and n. Further, $\mathbb{P}$ denotes the set of prime numbers. For $n = p_1^{e_1} \ldots p_m^{e_m}$ with prime number $p_1 < p_2 < \ldots p_m$ **Euler's function** $\varphi(n)$ denotes

$$|\mathbb{Z}_n^*| = (p_1 - 1){p_1}^{e_1 - 1} \ldots (p_m - 1){p_m}^{e_m - 1} =: \varphi(n) \ .$$

The set of *quadratic residues* $\mathbb{QR}_n$ and *non-quadratic residues* $\mathbb{NQR}_n$ is defined by

$$\begin{array}{rcl} \mathbb{QR}_n & := & \{i \in \mathbb{Z}_n^* \mid \exists a \in \mathbb{Z}_n \ a^2 \equiv i \pmod{n}\} \ , \\ \mathbb{NQR}_n & := & \mathbb{Z}_n^* \setminus \mathbb{QR}_n \ . \end{array}$$

The **Legendre-symbol** is defined for a prime number p and $a \in \mathbb{Z}_p^*$ as:

$$\left(\frac{a}{p}\right) := \begin{cases} +1 \ , & a \in \mathbb{QR}_p \ , \\ -1 \ , & a \in \mathbb{NQR}_p \ . \end{cases}$$

The **Jacobi-Legendre-symbol** enhances this symbol for all numbers $n \in \mathbb{N}$ and $a \in \mathbb{Z}_n^*$, where

$$\left(\frac{a}{n}\right) := \begin{cases} \left(\frac{a}{n}\right) \ , & n \text{ is prime number} \ , \\ \left(\frac{a}{p}\right) \cdot \left(\frac{a}{m}\right) \ , & n = p \cdot m \ \text{ and } p \text{ is prime number} \ . \end{cases}$$

Further, we define $\mathbb{Z}_m^\circ := \{z \in \mathbb{Z}^* \mid \left(\frac{z}{m}\right) = 1\}$ and $\mathbb{NQR}_m^\circ := \mathbb{Z}_m^\circ \cap \mathbb{NQR}_m$. The law of quadratic reciprocity allows an efficient computation of the Jacobi-Legendre-symbol, whereas in general it is assumed to be intractable to determine whether $x \in \mathbb{QR}_n$, if n factors into two large prime numbers and $\left(\frac{x}{n}\right) = 1$. This security assumption stated by Adleman is called **Intractability Assumption of Quadratic Residuosity (QRA)** [GoMi 82]:

> Let $0 < \epsilon < 1$. For each positive integer k let $C_{k,\epsilon}$ be the minimum size of circuits C that decide correctly quadratic residuosity mod n for a fraction ϵ of the k bit integers n. Then, for every $0 < \epsilon < 1$ and every polynomial Q there exists $\delta_{\epsilon,Q}$ such that $k > \delta_{\epsilon,Q}$ implies $C_{\epsilon,k} > Q(k)$.

For $x \in \mathbb{Z}_m^*$ we define the **quadratic residuosity** of x by $\mathbf{qr}(\boldsymbol{x}, \boldsymbol{m}) := 0$, if $x \in \mathbb{QR}_m$ and $\mathbf{qr}(\boldsymbol{x}, \boldsymbol{m}) := 1$ elsewhere.

3 The Toolbox

For the moment let us assume, that all communication can be received by all players and that each message's sender can be detected. Furthermore, all participants obey all syntactical restrictions of the protocols. Later on, we will discuss what happens if these assumptions are weakened.

3.1 Preparation

Each player $i \in \{1,\dots,k\}$ chooses large prime numbers $p_i, q_i \in \mathbb{P}$ forming i's secret key. His public key consists of $m_i = p_i \cdot q_i$ and $y_i \in \mathbb{NQR}^\circ_{m_i}$. By the following protocol, a player verifies the correct form of m and y [GHY 85]. It is sufficient to verify that m consists of some powers of two distinct prime numbers.

Create public key (m, y)

Step	Alice		All other players
1.	Alice chooses two secure primes p, q and computes $m = p \cdot q$. Alice chooses $y \in \mathbb{NQR}^\circ_m$.	$\xrightarrow{m,y}$	check $y \in \mathbb{Z}^\circ_m$.
2.	Alice proves Bob that m has only two primes factors [GHY 85]		
3.	Alice proves Bob that $y \in \mathbb{NQR}^\circ_m$		

The following number-theoretic fact enables a zero-knowledge proof of the number of prime factors of m.

Fact 1 *If m has k different odd prime factors, then it holds* $|\mathbb{QR}_m| = \frac{|\mathbb{Z}^*_m|}{2^k}$.

$\exists p, q \in \mathbb{P}, \nu, \eta \geq 1 \ : \ m = p^\nu q^\eta$ [GHY 85]

Step	Alice		Bob
1.			Bob checks, that $m^{j^{-1}}$ for $j \in \{1,\dots,\log m\}$ are not prime numbers.
2.		$\xleftarrow{r_1,\dots,r_s}$	Bob chooses security parameter s and s random numbers $r_1,\dots,r_s \in \mathbb{Z}^*_m$.
3.	For each $\ell \in \{1,\dots,s\}$		
3a.	If $r_\ell \in \mathbb{QR}_m$ then		
	Alice proves Bob that $r_\ell \in \mathbb{QR}_m$		
3b.	If $r_\ell \in \mathbb{NQR}_m$ then		
	Alice proves Bob that $r_\ell \in \mathbb{NQR}_m$		
4.			Bob accepts if $\lvert\mathbb{QR}_m \cap \{r_1,\dots,r_s\}\rvert \geq 3/8 \cdot s$.

Bob can sabotage Alice with 50% probability. For this, he can choose $r_1 \in \mathbb{NQR}_m$, with probability of $1/2$ and then he continuesly sends numbers $r_i = r^2 r_1$. Hence, it holds $r_i \in \mathbb{NQR}_m$. So, Alice has no chance to convince Bob and all witnesses that m has the desired factorization. We can avoid this situation if Bob sends only trusted random numbers that Alice and Bob have determined via coin flipping per telephone [Blum 81]. Under additional cryptographic assumption we have developed an efficient and secure multi-player protocol for flipping a large quantity of many random bits [Jako 98].

$\boldsymbol{y \in \mathbb{NQR}_m}$

Step	Alice		Bob
known	$p, q \in \mathbb{P}$, $m = pq$, $y \in \mathbb{NQR}^{\circ}_m$		$m \in \mathbb{N}$, $y \in ZC_m$
1.			Bob chooses security parameter s.
2.		$\xleftarrow{s, c_*}$	Bob chooses random numbers $r_1, \ldots, r_s \in \mathbb{Z}^*_m$. With 50 % probability Bob sends $c_i = yr_i^2 \bmod m$ or $c_i = r_i^2 \bmod m$.
3.	For all $i \in \{1, \ldots, s\}$ Alice computes $b_i = \mathrm{qr}(c_i, m)$.	$\xrightarrow{b_*}$	Bob checks for all $i \in \{1, \ldots, s\}$: $b_i = 0 \iff c_i = r_i^2 \bmod m$.

This protocol is not a zero-knowledge proof, since Bob gets the quadratic residue property of the numbers $c_1, \ldots, c_s$.

If hidden cards were already generated, the complete secret information of cards could be decoded. So, this protocol must only be used for establishing the public key. Later on, we show how to construct a protocol that is perfect zero-knowledge and may be used even during the game.

The following protocol proves that $t \in \mathbb{QR}_m$ using a zero-knowledge proof:

$\boldsymbol{t \in \mathbb{QR}_m}$

	Alice		Bob
knows	$p, q \in \mathbb{P}$, $m = pq$, $t \in \mathbb{QR}_m$		$m \in \mathbb{N}$, $t \in \mathbb{Z}^{\circ}_m$
1.		$\xleftarrow{s}$	Bob chooses security parameter s.
2.	Alice randomly chooses $r_1, \ldots, r_s \in \mathbb{Z}^*_m$ and computes $t_1, \ldots, t_s \in \mathbb{Z}^*_m$ such that $t \equiv r_i^2 \cdot s_i^2 \pmod{m}$. Let $R_i = r_i^2 \bmod m$, $S_i = s_i^2 \bmod m$.	$\xrightarrow{R_*, S_*}$ $\xleftarrow{X}$	Bob checks $\forall i \in \{1, \ldots, s\} : t \equiv R_i \cdot S_i \pmod{m}$ and chooses a random subset $X \subseteq \{1, \ldots, s\}$.
3.	For all $i \in \{1, \ldots, s\}$:		
3a.	If $i \in X$ then Alice sends r_i	$\xrightarrow{r_i}$	Bob checks $R_i = r_i^2 \bmod m$
3b.	If $i \notin X$ then Alice sends s_i	$\xrightarrow{r_i}$	Bob checks $S_i = s_i^2 \bmod m$

As soon as a trusted non quadratic residue $y \in \mathbb{NQR}^{\circ}_m$ is testified, we can use the following zero-knowledge proof for verifying $t \in \mathbb{NQR}^{\circ}_m$.

$\boldsymbol{t \in \mathbb{NQR}^{\circ}_m}$ **using** $\boldsymbol{y \in \mathbb{NQR}^{\circ}_m}$

	Alice		Bob
knows	$p, q \in \mathbb{P}$, $m = pq$, $t, y \in \mathbb{NQR}^{\circ}_m$		$m \in \mathbb{N}$, $t \in \mathbb{Z}^{\circ}_m$, $y \in \mathbb{NQR}^{\circ}_m$
1.	Alice computes $r \in \mathbb{QR}_m$ such that $t \equiv y \cdot r \pmod{m}$.	$\xrightarrow{r}$	Bob checks $t \equiv y \cdot r \pmod{m}$.
2.	$\boxed{\text{Alice proves Bob that } r \in \mathbb{QR}_m}$		

3.2 Masking Numbers

The data structure for cards consists of numbers $z \in \mathbb{Z}^{\circ}_m$, which are called the reverse side of a card. Alice computes a new reverse side by **masking** each entry z by

$$z' = z \cdot r^2 \cdot y^c \bmod m_i$$

for some $r \in \mathbb{Z}_m^*$, $c \in \{0,1\}$. We denote this relationship between z and z' by $\boldsymbol{z \hookrightarrow z'}$. Using the factorization of m given z, z' one can always establish such a relationship, and if one can establish such a relationship for given z, z', this gives the factorization of m:

Lemma 1 *Let m be a product of two prime numbers. A polynomial time bounded algorihm that computes for given $z, z' \in \mathbb{Z}_m^*, y \in \mathbb{NQR}_m$ a number $r \in \mathbb{Z}_m^*$ such that $z' = z \cdot r^2 \cdot y^b \bmod m$ for $b \in \{0,1\}$ can be used to compute the factorization of m in expected polynomial time.*

Proof: First note that every square $s \in \mathbb{Z}_m^*$ has four roots $r_1, r_2, -r_1, -r_2 \in \mathbb{Z}_m^*$, where $\gcd(r_1 + r_2, m)$ is a non-trivial factor of m. Further, note that $z' \cdot z^{-1} \cdot y^{-b} \in \mathbb{QR}_m$ has square root r. So, we choose random numbers $r, z \in \mathbb{Z}_m^*$, and a random bit $b \in \{0,1\}$ and compute $z = z \cdot r^2 \cdot y^b \bmod m$. Assume that such an algorithm exists then it computes $r_1 = r$ and $-r_1$ with probability $1/2$ and r_2 or $-r_2$ with probability $1/2$. The latter case gives the prime factorization. ∎

Without the knowledge of factorization of m one cannot relate x to z in general, but can build arbitrary large sets of numbers masking some given x, z to $x \hookrightarrow x_1 \hookrightarrow x_2, \ldots$; $z \hookrightarrow z_1 \hookrightarrow z_2, \ldots$. These sets form polynomial-time sampleable equivalence classes: The mask operation is reflexive $z \hookrightarrow z$; commutative $z_1 \hookrightarrow z_2 \iff z_2 \hookrightarrow z_1$; and transitive $z_1 \hookrightarrow z_2 \hookrightarrow z_3 \iff z_1 \hookrightarrow z_3$. So, these sets x_* and z_* remain disjunct, so long as Alice cannot factor m in expected polynomial time. To every other player Alice's built relationship is perfectly hidden, since the mask operation produces uniform distributed numbers over $\mathbb{Z}_m^\circ$. Nevertheless, Alice can prove the mask relationship by the following zero-knowledge proof.

Proof that $\boldsymbol{z \hookrightarrow z'}$

Step	Alice		Bob
knows	$m \in \mathbb{N}$, $z, z' \in \mathbb{Z}_m^*$, $y \in \mathbb{NQR}_m^\circ$, $r \in \mathbb{Z}_m^*$, $b \in \{0,1\}$, $$z' = z \cdot r^2 \cdot y^b \bmod m$$		$m \in \mathbb{N}$, $z, z' \in \mathbb{Z}_m^*$, $y \in \mathbb{NQR}_m^\circ$
1.		$\xleftarrow{s}$	chooses s.
2.	Alice computes $t_1, \ldots, t_n$ such that $\forall i \in \{1, \ldots s\}$: $z' \hookrightarrow t_i$, i.e. $$t_i = z' \cdot r_i^2 \cdot y^{b_i} \bmod m$$ for random $r_i \in \mathbb{Z}_m^*$, $b_i \in \{0,1\}$.	$\xrightarrow{t_*}$ $\xleftarrow{X}$	Bob chooses a random subset $X \subseteq \{1, \ldots, s\}$.
3.	For all $i \in \{1, \ldots, s\}$:		
3a.	If $i \in X$ then Alice publishes the secret parameters of $z' \hookrightarrow t_i$	$\xrightarrow{r_i, b_i}$	checks $z' \hookrightarrow t_i$
3b.	If $i \notin X$ then Alice publishes the secret parameters of $z \hookrightarrow t_i$, namely $b_i' = b \oplus b_i$ and $r_i' = r_i \cdot r \cdot y^{\lfloor (b+b_i)/2 \rfloor}$	$\xrightarrow{r_i', b_i'}$	checks $z \hookrightarrow t_i$

Lemma 2 *If Alice cannot deduce an $r \in \mathbb{Z}_m^*$ and $b \in \{0,1\}$ such that $z' = z \cdot r^2 \cdot y^b \bmod m$ ($z \hookrightarrow z'$) then Bob can convict Alice with probability $1 - 2^s$ using this protocol. Moreover, this protocol is perfect zero-knowledge.*

Proof:

- Perfect zero-knowledge: We show that an uninvolved party M can generate in polynomial time the same probability distribution of the communication as by the protocol. For this,

  1. M randomly chooses X.
  2. M randomly chooses $r_i \in \mathbb{Z}_m^*$, $b_i \in \{0,1\}$ for $i \in \{1, \ldots, s\}$.
  3. If $\ell \in X$ then M computes $t_\ell = z' \cdot r_\ell^2 \cdot y^{b_i} \bmod m$.

4. If $\ell \notin X$ then M computes $t_\ell = z \cdot r_\ell^2 \cdot y^{b_i} \bmod m$.

- Security: Assume that $z \not\hookrightarrow z'$ from Alice's point of view. Then either one of the following three cases hold:

  1. $z \hookrightarrow t_\ell$ and $z' \not\hookrightarrow t_\ell$
  2. $z \not\hookrightarrow t_\ell$ and $z' \hookrightarrow t_\ell$
  3. $z \not\hookrightarrow t_\ell$ and $z' \not\hookrightarrow t_\ell$

  With at least 50% probability Bob can convict Alice in any of these cases, since Alice has to publish the secret parameters of $z \hookrightarrow t_\ell$ or $z' \hookrightarrow t_\ell$.

■

A special case of masking is $1 \hookrightarrow z$, i.e. Alice proves Bob that she knows $r \in \mathbb{Z}_m^\circ$ and $b \in \{0,1\}$ such that $z = y^b \cdot r^2 \bmod m$.

Proof of $1 \hookrightarrow t$

	Alice		Bob
knows	$m \in \mathbb{N}$, $y \in \mathbb{NQR}_m^\circ$, $r \in \mathbb{Z}_m^*$, $b \in \{0,1\}$, $t = y^b r^2 \bmod m$, $r \in \mathbb{Z}_m^*$, $b \in \{0,1\}$		$m \in \mathbb{N}$, $y \in \mathbb{NQR}_m^\circ$, $t \in \mathbb{Z}_m^*$
1.		$\xleftarrow{s}$	Bob chooses security parameter s.
2.	Alice randomly chooses $r_1, \ldots, r_s \in \mathbb{Z}_m^*; b_1, \ldots, b_s \in \{0,1\}$. Then she computes $t_1, \ldots, t_s \in \mathbb{Z}_m^*, c_1, \ldots, c_s \in \{0,1\}$ such that $t \equiv r_i^2 \cdot s_i^2 \cdot y^{b_i + c_i} \pmod m$. Let $R_i = r_i^2 y^{b_i} \bmod m$, $S_i = s_i^2 y^{c_i} \bmod m$.	$\xrightarrow{R_*, S_*}$	Bob checks $\forall i \in \{1, \ldots, s\} : t \equiv R_i \cdot S_i \pmod m$ and chooses a random subset $X \subseteq \{1, \ldots, s\}$.
3.	For all $i \in \{1, \ldots, s\}$:		
3a.	If $i \in X$ then Alice sends r_i, b_i	$\xrightarrow{r_i}$	Bob checks $R_i = r_i^2 \cdot y^{b_i} \bmod m$
3b.	If $i \notin X$ then Alice sends s_i, c_i	$\xrightarrow{r_i}$	Bob checks $S_i = s_i^2 \cdot y^{c_i} \bmod m$

A zero-knowledge proof of non-quadratic residuosity Note that for the standard proof of $t \in \mathbb{NQR}_m^\circ$ either the prover has to publish the quadratic residue property of some numbers or all players have to use the public key information $y \in \mathbb{NQR}_m^\circ$. The protocol of $1 \hookrightarrow z$ helps to overcome these disadvantages and enables us to decrease the error probability of $y \in \mathbb{NQR}_m^\circ$ even during the game.

Zero-knowledge proof of $y \in \mathbb{NQR}_m$

Step	Alice		Bob
known	$p, q \in \mathbb{P}$, $m = pq$, $y \in \mathbb{NQR}^\circ_m$		$m \in \mathbb{N}$, $y \in ZC_m$
1.			Bob chooses security parameter s.
2.		$\xleftarrow{s}$ $\xleftarrow{t_*}$	Bob chooses random numbers $r_1, \ldots, r_s \in \mathbb{Z}^*_m$ and chooses a random subset $X \subseteq \{1, \ldots, s\}$. For all $\ell \in X$ Bob computes $t_\ell = r_\ell^2 \bmod m$ and for all $\ell \notin X$ $t_\ell = r_\ell^2 \cdot y \bmod m$.
3.	For all $\ell \in \{1, \ldots, s\}$		
3a.	$\boxed{\text{Bob proves Alice that } 1 \hookrightarrow t_\ell}$		
4.	Alice computes $Y = \{i \mid t_i \in \mathbb{QR}_m\}$.	$\xrightarrow{Y}$	Bob checks $X = Y$.

3.3 The Data Structure for Cards

Every card is represented by an encoding, called **reverse** which is known to all players. This information does not help (if QRA holds) to decode unless all players allow it a player. To keep the card secret the information is spread among the players secret coding schemes (see fig. 2). The information on the reverse is encoded by all the players' cryptographic schemes. Only if all players reveal their encoding the type of a card can be read.

Let T be the number of different types of cards, k the number of players and $w = \lceil \log_2 T \rceil$. A reverse consists of $k \cdot w$ numbers $(z_{1,1}, \ldots, z_{k,w})$. For $i \in \{1, \ldots, k\}$, $j \in \{1, \ldots, k\}$, it holds $z_{i,j} \in \mathbb{Z}^\circ_{m_i}$. Every number $z_{i,j}$ encrypts a bit. This bit can be computed by

$$b_{i,j} = \mathrm{qr}(z_{i,j}, m_i) = \begin{cases} 0 \ , & z_{i,j} \in \mathbb{QR}_{m_i} \ , \\ 1 \ , & \text{else.} \end{cases} \quad .$$

The type t of a card can be computed by the term:

$$t = 1 + \sum_{j=1}^{w} 2^{j-1} \cdot \bigoplus_{i=1}^{k} b_{i,j} .$$

The quadratic residuosity of numbers solely encodes the type of a card. So, throughout all protocols no information about quadratic residuosity may be published unless there is a proof that the numbers are related by consecutive mask operations. Then, the owners of the secret keys do not publish new information that could have been produced by public-key information if all the players played honestly.

Creation of an open card For a card of type t a player computes the binary representation $b_1, \ldots, b_k \in \{0, 1\}$ of t and publishes

$$C = ((y_1^{b_1}, \ldots, y_1^{b_k}), (1, \ldots, 1), \ldots, (1, \ldots, 1))$$

using only public known information.

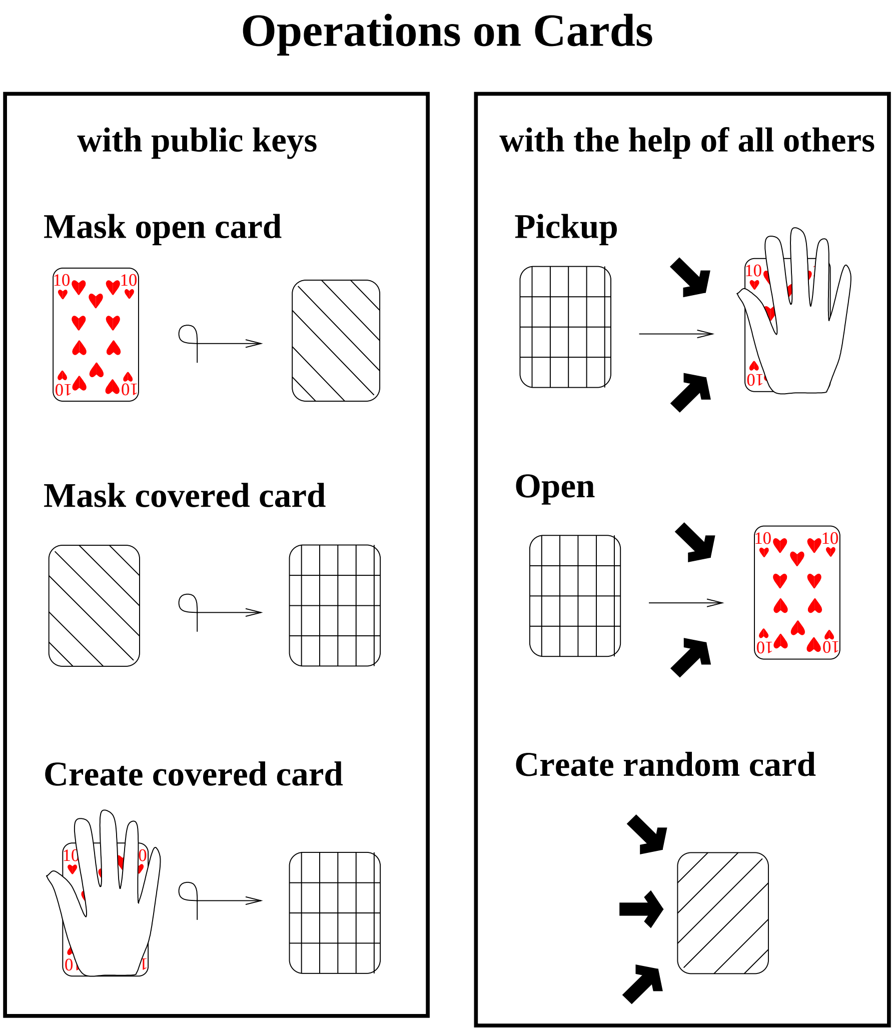

Figure 1: Some available operations on cards.

Mask a Card There is a public available algorithm that computes for a given card another card uniformly distributed over the set of all cards of that type. We denote $\boldsymbol{C} \looparrowright \boldsymbol{C'}$ for the relationship between original card C and new card C'.

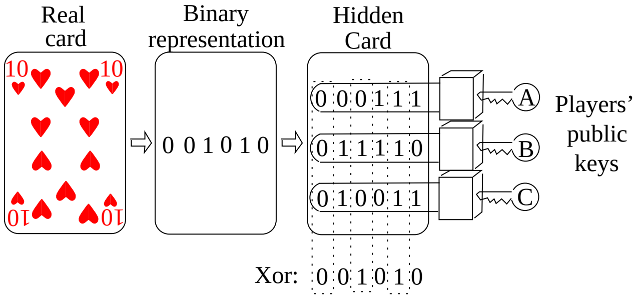

Figure 2: Data structure used for cards. Each bit of the binary representation is encoded by three bits which give the original bit by an exclusive or.

Mask a card $C \looparrowright C'$

1. For every card $C \in D$ Alice flips $kw - w$ random bits $(b_{2,1}, \ldots, b_{2,w}), \ldots, (b_{k,1}, \ldots, b_{k,w})$ and kw random numbers $(r_{i,j})_{i,j \in \{1,\ldots,k\} \times \{1,\ldots,w\}} \in (\mathbb{Z}^*_{m_1})^w \times \cdots \times (\mathbb{Z}^*_{m_k})^w$.
2. Alice computes for every card C and for all $j \in \{1, \ldots, w\}$:
$$b_{1,j} = \bigoplus_{i \in \{2,\ldots,k\}} b_{i,j} \ .$$
3. Now Alice computes for a given card $C = (z_{*,*})$ the new reverse $C' = (z'_{*,*})$. The new values z'_i are computed for $i \in \{1, \ldots, k\}$, $j \in \{1, \ldots, w\}$ by
$$z'_{i,j} = z_{i,j} \cdot r^2_{i,j} \cdot y_i^{b_{i,j}} \bmod m_i \ .$$
4. Alice proves all other players that $C \looparrowright C'$ is correct.

Theorem 1 *The mask operation generates cards which are distributed uniformly over all cards with the same type. No coalition can decode a masked card C' if the type of original card C is not known. Computing all secret parameters of a mask operation is computationally equivalent to computing the factorization of all m_i.*

Proof:

1. The mask operation generates a uniform probability distribution:

   Note that the mask operation for numbers maps to all numbers of $\mathbb{Z}^\circ_m$ with same probability. Further, the mask-card operation is symmetric w.r.t. to the players' encoding such that all binary information of type t occurs with same probability.

2. No coalition can decode a masked card $C' = z'_{*,*}$ against player.

   Assume that all players form a coalition against a player. Of course they know the binary information $\mathrm{qr}(z_{i,j})$ of card C and $\mathrm{qr}(z'_{i,j})$ of C' for $i > 1$ and so they can compute all the secret binary information involved in a masking operation. Nevertheless, since the decisive information $\mathrm{qr}(z_{1,j})$ stays unknown the masked card C' cannot be decoded, too.

3. Computing all secret parameters of a mask operation between two given cards of same type is computationally equivalent to compute the factorization all m_*.

   We have seen that for given C, C' a coalition of all but one players can compute the binary information of masking. Furthermore, they can compute all secret information of $z_{i,j} \hookrightarrow z_{i,j}$ for $i > 1$ since they know the factorization of m_i . But as missing information the square roots of $z_{1,j} \hookrightarrow z_{1,j}$ remain. The claim follows by Lemma 1.

∎

Furthermore, a mask operation for cards forms an equivalence class: Masking is reflexive, transitive and commutative. These facts help us to create the following zero-knowledge proof of a correct masking. Given random cards C, C' of the same type no other player than the creator can deduce all the secret information of $C \looparrowright C'$ since $w \cdot k$ masking operations of k cryptographic public-key systems are involved. But every player can mask a card and if all players do this, none can deduce all secret parameters.

Proof that $C \looparrowright C'$

Step	Alice		All other players
	All know the public keys and cards C, C		
	Alice knows secret parameters of $C \looparrowright C'$		
1.		$\xleftarrow{s}$	choose security parameter s.
2.	Alice computes $C_1, \ldots, C_s$ such that $\forall \ell \in \{1, \ldots s\} \ : C' \looparrowright C_\ell$.	$\xrightarrow{C_*}$ $\xleftarrow{X}$	choose a random subset $X \subseteq \{1, \ldots, s\}$.
3.	For all $\ell \in \{1, \ldots, s\}$:		
3a.	If $\ell \in X$ Alice publishes the secret parameters of $C' \looparrowright C_\ell$	$\xrightarrow{r_*, b_*}$	checks $C' \looparrowright C_\ell$
3b.	If $\ell \notin X$: Alice publishes the secret parameters of $C \looparrowright C_\ell$	$\xrightarrow{r'_*, b'_*}$	checks $C \looparrowright C_\ell$

Theorem 2 *This protocol is perfect zero-knowledge and proves to all other players the correctness of the masking-operation with an error probability of at most* 2^{-s}.

Proof:

1. We can produce the same probability distribution as the communications only with the knowledge of C,C', and the public keys as follows.

   We choose a random subset $X \subseteq \{1, \ldots, s\}$. Then we compute $C' \looparrowright C$ for all $\ell \in X$ and $C \looparrowright C_\ell$ elsewhere. In both cases we publish the secret mask parameters.

2. For the error probability note that $C' \looparrowright C_\ell$ and $C \looparrowright C_\ell$ imply $C \looparrowright C'$, since the mask operation is commutative and transitive. Therefore in each round all other player have a 50% chance to convict Alice if $C \not\looparrowright C'$.

∎

Creation of private card is another application of the $1 \hookrightarrow z$-protocol.

Create a private card

	Alice		other players
knows	all public keys and her secret key		know all public keys
1.	Alice creates an open card C of type t for internal use only.		
2.	Alice masks $C \looparrowright C' = (z_{*,*})$	$\xrightarrow{C'}$	
3.	Alice proves $1 \hookrightarrow z_{i,j}$ for all $i \in \{1, \ldots, k\}$, $j, \in \{1, \ldots, m\}$.		

The last step ensures that Alice does not use numbers for the card which give information about other cards.

Theorem 3 *The encoding of such a generate private card cannot be decoded by all other players, if QRA holds.*

Proof: The j-th bit of the type of the card is given by the parity with $\mathrm{qr}(z_{1,j})$, which is chosen randomly. So, a coalition of all but one players has no chance to compute a bit if QRA holds. ∎

Generation of covered random cards The players want to generate a covered card with random type of $\{1,\dots,T\}$. For the beginning assume that $T = 2^w$.

Create covered random card

1. Every player $i \in \{1,\dots,k\}$ randomly chooses $z_{i,j} \in \mathbb{Z}^\circ_{m_i}$ for $j \in \{1,\dots,w\}$ and publishes it.
2. The card C is given by $z_{*,*}$

In this protocol the proof of correct masking is obsolete, since each player only uses his key. Although every player uses only public key information it is not desirable to allow other players to create one's encoding, simply because the card is not trustable random. But also when the card is picked up or opened, this player could achieve information about the quadratic residuosity of numbers on the reverse sides of other cards.

Suppose that T is no power of 2. Then the situation becomes more complicated. We will choose w such $2^{w-1} \leq T \leq 2^w$ and use the original protocol for the creation of the covered card. If Alice picks up a card she sees whether its type is out of range. Then Alice opens this cards and may get a new covered card using the protocol shown above. Now she privatizes this card and this continues as long as a correct card appears. Of course the expected number of rounds is less than 2 and Alice gets her card considerably faster than using a mix-protocol of a deck with T cards shown below.

This way all players may compute trustable random numbers in every range, e.g. simulating flipping coins or cubes: They compute a covered random card and then open it (see below).

Pick up a card In the following protocol a player gets the decoding of a covered card. All other players eaves-dropping the protocol gain no information, since the crucial information stays encoded:

Pick up a card

	Alice	Bob	Charly
	know all public keys and the covered card $C = (z_{*,*})$		
	knows her secret key p_1, q_1	his secret key p_2, q_2	his secret key p_3, q_3
1.		Bob sends the information $\mathrm{qr}(z_{2,j}, m_2)$ for $j \in \{1,\dots,w\}$	Charly sends the information $\mathrm{qr}(z_{3,j}, m_3)$ for $j \in \{1,\dots,w\}$
2.		Bob proves Alice the correctness.	Charly proves correctness.

All used proofs used are perfect zero-knowledge. But by sending the quadratic residuosity there is a way for Alice to get information that may not be designated for her.

Consider Alice creating the card C without proving the correct form w.r.t. masking. So, she can get secret information about another covered card D. For this, she may directly copy all entries of D to C or she may mask the entries z using $z \hookrightarrow z'$. So, Alice can read both cards C and D using one pick-up operation. Therefore, the proof of correct masking a card is essential.

Theorem 4 *If a card C is the result of a create-open-card, create-random-covered-card, or create-private-card operation followed by a sequence of mask-card operations, then the pick-up operation does only give further information as the type of the card picked up. All players besides Alice cannot extract the type of the card picked up unless they already know it.*

Proof:

1. Consider a player Z getting all public and secret parameters of creation and masking a card. Further Z gets the public key information. We prove the first claim by showing that this Z is now able to compute the same information published in the pick-up procedure.

Let C_0 be the at first published card and C the card that is picked up. Further, let $C_0 \looparrowright C_1 \looparrowright \cdots \looparrowright C$ be the masking steps. These steps can be contracted to $C_0 \looparrowright C$ by Z such that he knows the secret parameters of this masking. There are three cases:

(a) C_0 was an open card. Then, Z can compute all secret parameters in C and therefore $\mathrm{qr}(z_{i,j})$ for $i \in \{1, \ldots, k\}$, $j \in \{1, \ldots, w\}$.

(b) C_0 was a private card. Since Z has the private mask parameters this case can be reduced to the above situation.

(c) C_0 was a covered random card. Again Z has collected all private mask parameters, which gives the same situation as for a private card.

This completes the proof since all mask operation use only public key information and therefore the overall operation of hiding and picking up a card is an operation that Z could have done by himself with the same probability distribution.

2. Note that the security of a covered card does not depend on the mask operations but the missing knowledge of the quadratic residuosity of the first card. The decisive secure part is the quadratic residuosity w.r.t. the module of the player picking up the cards, which is the only one that an all but one player coalition cannot decode.

∎

Open a private card This protocol is similar to the last stated. Note that Alice cannot prove the correctness of the other players' encodings. This has to be done by them.

Open a card

	Alice	Bob	Charly
	know all public keys and the covered card $C = (z_{*,*})$		
	knows her secret key p_1, q_1	his secret key p_2, q_2	his secret key p_3, q_3
1.	Alice sends all other players $\mathrm{qr}(z_{i,j}, m_i)$ for $i \in \{1, \ldots, k\}$, $j \in \{1, \ldots, w\}$		
2.	Alice proves the correctness of $\mathrm{qr}(z_{1,j}, m_1)$ for $j \in \{1, \ldots, w\}$	Bob proves the correctness of $\mathrm{qr}(z_{2,j}, m_2)$ for $j \in \{1, \ldots, w\}$	Alice proves the correctness of $\mathrm{qr}(z_{3,j}, m_3)$ for $j \in \{1, \ldots, w\}$

Again it is crucial that the card was created by the mask operation (that should have been proved before this step). If not, some player (the one that at last should have masked the card) may get some additional information.

Theorem 5 *If a card C is the result of a create-open-card, create-random-covered-card, or create-private-card operation followed by a sequence of mask-card operations, then the open operation does only give information about C.*

Proof: analogously to Theorem 4. ∎

3.4 Operations on the Deck

A deck is modelled by a tuple of cards. Stacks are not necessarily disjunct subsets of the deck. In the beginning of the game some player may **create the deck**. This operation corresponds to a number of creations of open cards shown above. Contrary to the protocols known so far, it is always possible to include a new deck later on.

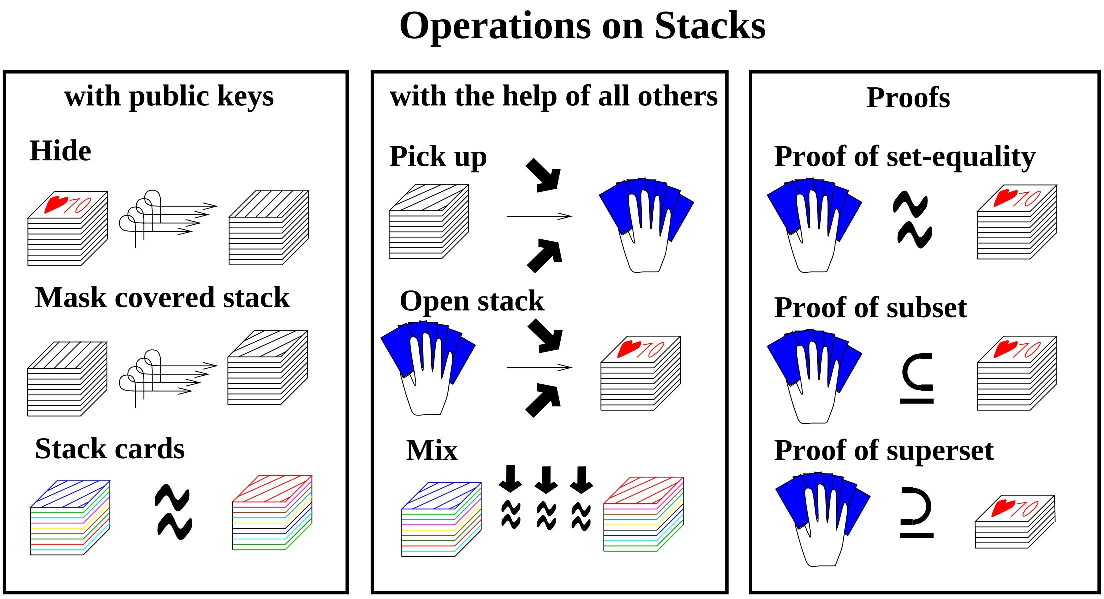

Figure 3: Some available operations on stacks.

Stacking the cards Alice stacks a set of d cards S, i.e. she masks all cards and applies a secret permutation σ to the stack.

Stack the cards S to S': $S \approx S'$

1. Alice masks each card $C_i \in S$: $C_i \looparrowright C'_i$, for $i \in \{1, \ldots, d\}$.
2. Alice changes the position of all cards using her secret permutation σ and publishes the new covered stack $S' = (C'_{\sigma(1)}, \ldots, C'_{\sigma(d)})$.
3. Alice proves all players the correctness of $S \approx S'$.

The secret parameters of this permutation are called $\boldsymbol{P_\sigma}$, similary to the ANDOS-scheme [BCR 87]. W.r.t ANDOS additional information is encoded that allows the publication of the new permuted stack:

$$P_\sigma = \begin{pmatrix} \sigma(1) & b_{1,(1,1)} & \cdots & b_{1,(k,w)} & r_{1,(1,1)} & \cdots & r_{1,(k,w)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \sigma(d) & b_{d,(1,1)} & \cdots & b_{d,(k,w)} & r_{d,(1,1)} & \cdots & r_{d,(k,w)} \end{pmatrix}.$$

Alice's transformation of the stack is denoted by $\boldsymbol{S \overset{P_\sigma}{\approx} S'}$. Alice does not use her secrets p and q for this operation. So, even an uninvolved party can create and prove a valid P_σ only by using public available information. Since for the open- and pick-up-protocols it is essential that all entries of the cards are masked correctly Alice has to prove the correctness of her stacking-operation.

For given P_σ and P_π we compute $P_{\pi\circ\sigma}$, where $S \overset{P_\sigma}{\approx} S' \overset{P_\pi}{\approx} S'' \iff S \overset{P_{\pi\circ\sigma}}{\approx} S''$. Let P_π given by

$$P_\pi = \begin{pmatrix} \pi(1) & c_{1,(1,1)} & \cdots & c_{1,(k,w)} & s_{1,(1,1)} & \cdots & s_{1,(k,w)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \pi(d) & c_{d,(1,1)} & \cdots & c_{d,(k,w)} & s_{d,(1,1)} & \cdots & s_{d,(k,w)} \end{pmatrix}.$$

Then $P_{\pi\circ\sigma}$ is given by

$$P_{\pi\circ\sigma} = \begin{pmatrix} \pi(\sigma(1)) & b_{\sigma(1),(1,1)} \oplus c_{\sigma(1),(k,w)} & \cdots & b_{\sigma(1),(k,w)} \oplus c_{1,(k,w)} & t_{1,(1,1)} & \cdots & t_{1,(k,w)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \pi(\sigma(d)) & b_{\sigma(d),(1,1)} \oplus c_{d,(1,1)} & \cdots & b_{\sigma(d),(k,w)} \oplus c_{d,(k,w)} & t_{d,(1,1)} & \cdots & t_{d,(k,w)} \end{pmatrix},$$

where

$$t_{\nu,(i,j)} = \left(r_{\sigma(\nu),(i,j)} \cdot s_{\nu,(i,j)} \cdot y_i^{\lfloor (b_{\sigma(nu),(i,j)} + c_{\nu,(i,j)})/2 \rfloor}\right) \bmod m_i$$

On the other hand, the knowledge of $P_{\pi\circ\sigma}$ and P_π enables the computation of P_σ using only public keys. In the following protocol Alice proves the correctness of her P_σ, by proving that she properly masks the cards analogously as above in the mask-operation.

Proof of P_σ

Step	Alice		All other players
	Alice has P_σ kept in secret, such that $S \overset{P_\sigma}{\approx} S'$.		know S, S'
1.		$\xleftarrow{s}$	agree on a security parameter s.
2.	Alice chooses s permutations $\sigma_1, \ldots, \sigma_s$ over $\{1, \ldots, t\}$ and corresponding $P_{\sigma_1}, \ldots, P_{\sigma_s}$. Using this information, Alice creates new permutations of the stack $S' \overset{P_{\sigma_1}}{\approx} S_1, \ldots, S' \overset{P_{\sigma_s}}{\approx} S_s$ and publishes them.	$\xrightarrow{S_*}$ $\xleftarrow{X}$	agree on a random subset $X \subseteq \{1, \ldots, s\}$.
3.	For all $\ell \in \{1, \ldots, s\}$:		
3a.	If $\ell \in X$ Alice publishes P_{σ_ℓ}.	$\xrightarrow{P_{\sigma_\ell}}$	check whether $S' \overset{P_{\sigma_\ell}}{\approx} S_\ell$.
3b.	If $\ell \notin X$ Alice publishes $P_{\sigma_\ell\circ\sigma}$.	$\xrightarrow{P_{\sigma_\ell\circ\sigma}}$	check whether $S \overset{P_{\sigma_\ell\circ\sigma}}{\approx} S_\ell$.

Theorem 6 *The stacking operation is perfect zero-knowledge and can be performed by only using the public keys. It performs a permutation on the given stack which stays secret to all other players. A series of stacking operations followed by a pick-up or open-operation does not reveal any additional information but the type of the addressed card and its former place in the stack.*

Proof:

1. Proof of P_σ is perfect zero-knowledge:

   Generate a random subset $X \subseteq \{1, \ldots, s\}$. For all $\ell \in \{1, \ldots, s\}$: if $\ell \in X$ then compute $S' \approx S_\ell$ and else compute $S \approx S_\ell$. In both cases output the secret parameters.

2. It performs a permutations on the given stack which stays secret to all other players:

   Let S be the original stack and S' the output of the operation. Assume that the stacking operation is not a permutation. Then a type t exists such that it occurs n times in S and n' times in S', such that $n \neq n'$. Let n_ℓ be the number of occurences of t in S_ℓ. For n_ℓ there are three cases:

   (a) $n = n_\ell$, $n' \neq n_\ell$

   (b) $n \neq n_\ell$, $n' = n_\ell$

   (c) $n \neq n_\ell$, $n' \neq n_\ell$

   Since Alice has to prove $S \approx S_\ell$ or $S' \approx S_\ell$ by publishing all secret parameters without knowing which one in advance. All other players can convict Alice with at least a 50% probability in each of the s rounds.

   Since the masking operation produces a uniform probability distribution over all cards of the same type. That part of the cards in the new stack, which a all-but-one coalition of players can decode and analyze, is produced by the same uniform probability distribution.

3. Assume that after a pick-up procedure all players who performed a stacking procedure publish a trace of the picked-up or opened card through all permutations. Then, we get a sequence $C_0 \looparrowright C_1 \looparrowright \cdots \looparrowright C$. Now, the claim follows by Theorem 4 and 5.

∎

Mixing a stack The best way to mix a stack is to let each of the players stack the cards. Such a stack may consist of private, open, or random covered cards and this operation may be performed in every state of the game whenever the rules allow this operation.

Mix a stack

	Alice	Bob	Charly
	They know all public keys and the covered stack S		
1.	stacks S to S'.	stacks S' to S''.	stacks S'' to the resulting stack S'''.
2.	Alice proves $S \approx S'$		
3.	Bob proves $S' \approx S''$		
4.	Charlie proves $S'' \approx S'''$		

Drawing a card from a private or open stack If some player wants to draw a card from a stack, at first he informs all players which card he is going to take by quoting its encoded side. Then the above mentioned "pick up a card"-protocol follows. If he wants to draw a card from the hand of a player, according to the rules of the games the other player may stack his cards before he gets a card picked away. If all other players insist on a random card they can determine a random number which gives the position (shown above).

Set-comparisons of private and public stacks Consider a private stack S and a stack of open cards U. A player can easily prove that S is a permuted version of U by constructing a corresponding P_σ and prove its correctness. If a player wants to prove that a private stack S is a subset of an open stack U without revealing S he creates the missing cards $U \setminus S$ and mixes them into S getting S'. Now he proves the equality between S' and U:

Proof of $S \subseteq U$

Step	Alice		All other players
known	Alice has private stack S. U is an open stack.		open stack U, covered stack S.
1.	Alice creates hidden cards $A = U \setminus S$. and combines S and A in this ordering to stack S'	$\xrightarrow{S'}$	verify that the reverse sides of S are on top of S'
2.	Alice stacks U to S'		
3.	Alice proves that $U \approx S'$		

Proof of $S \supseteq U$

Step	Alice		All other players
known	private stack S, open stack U.		open stack U, reverse sides of S.
1.	Alice creates hidden cards $A = S \setminus U$. and combines U and A in this ordering to stack U'	$\xrightarrow{S'}$	verify that the reverse sides of U are on top of U'
2.	Alice stacks U' to S		
3.	Alice proves that $U' \approx S$		

For the following protocol we assume that all types of cards only appear once and thus identify a card:

Proof of $S \cap U = \emptyset$

Step	Alice		All other players
known	Alice has private stack S. U is an open stack, set of all types $\mathcal{U}$.		open stack U, reverse sides of S, set of all types $\mathcal{U}$.
1.	Alice combines S, U and the created hidden cards $A = \mathcal{U} \backslash (U \cup S)$ to stack S'.	$\xrightarrow{S'}$	verify that U, S are on top of S'
2.	Alice stacks S' to D, which is the deck.		
3.	$\boxed{\text{Alice proves that } S' \approx D}$		

If a type of a cards may appear more than once, we use multisets as a representation of all cards and insert a card of a type $|S|$ times in the set of all types. The rest of the protocol is analogously.

Rule control In some games like "Skat" or "Schafkopf" there are some restrictments according to cards that a player may lay on the open stack, e.g. on a card of spades only a card of different type may be put, if the player cannot serve, i.e. has no spades in his private stack. Normally, such rules cannot be controlled during the game without a referee. In fact our scheme can prevent the breaking of such a rule just in time.

More formally, let $U \subseteq \{1, \ldots, T\}$ be a set of types of cards. Let S denote the private stack of Alice. The following protocol proves that Alice chooses a card $C \in S$ such that if $U \cap S \neq \emptyset$ then $C \in U$ without publishing the type of C and whether $U \cap S \neq \emptyset$.

Alice serves card C of type U if possible

Step	Alice		All other players
known	Alice has private stack S with card $C \in S$, U is an open stack, U_1 is the first card of U, H is an arbitrary subset of $D \setminus U$ of size $\lvert S \rvert$.		open stacks H,U, reverse sides of S, C.
1.		$\xleftarrow{s}$	agree on a security parameter s.
2.	Alice generates a random subset $X \subseteq \{1, \ldots, s\}$.		
3.	For all $\ell \in \{1, \ldots, s\}$:		
3a.	If $\ell \in X$ then Alice generates $S \approx S^1_\ell$, $U_1 \looparrowright C^1_\ell$, $H \approx S^2_\ell$, $C \looparrowright C^2_\ell$.	$\xrightarrow{S^1_\ell, S^2_\ell}$ $\xrightarrow{C^1_\ell, C^2_\ell}$	
3b.	If $\ell \notin X$ then Alice generates $S \approx S^2_\ell$, $U_1 \looparrowright C^2_\ell$, $H \approx S^1_\ell$, $C \looparrowright C^1_\ell$.	$\xrightarrow{S^1_\ell, S^2_\ell}$ $\xrightarrow{C^1_\ell, C^2_\ell}$	
4.		$\xleftarrow{Y}$	All players agree on a random subset $Y \subseteq \{1, \ldots, s\}$.
5.	For all $\ell \in \{1, \ldots, s\}$:		
5a.	If $\ell \in X \cap Y$ then $\boxed{\text{Alice proves that } S^1_\ell \approx S,\ S^2_\ell \approx H,\ C \looparrowright C^2_\ell, U_1 \looparrowright C^1_\ell}$		
5b.	If $\ell \in Y \setminus X$ then $\boxed{\text{Alice proves that } S^2_\ell \approx S,\ S^1_\ell \approx H,\ C \looparrowright C^1_\ell, U_1 \looparrowright C^2_\ell}$		
5c.	If $\ell \in X \setminus Y$ then $\boxed{\text{Alice proves that } S^2_\ell \cap U = \emptyset,\ \{C^1_\ell\} \subseteq U}$		
5c.	If $\ell \notin X \cup Y$ then $\boxed{\text{Alice proves that } S^1_\ell \cap U = \emptyset,\ \{C^2_\ell\} \subseteq U}$		

Split the cards Splitting the cards means that a player chooses a secret number of cards of the top of a stack and places them in this order to the end of the stack. This operation corresponds to a cyclic shift. Since all player can see the card by their reverse sides we cannot simulate this operation in its original setting — the secret shift parameter could be seen by all the participants. Note that cyclic shifts are a subset of permutation that are closed against iteration. The basic idea for a protocol for proving the correctness of a secret cyclic shift, is an adaption of the stacking operation:

Secret cyclic shift

1. Alice masks every card C_i of S: $C_i \looparrowright C'_i$.
2. Alice performs her secret cyclic shift operation. For this, she removes c cards from top of the stack and places them at the bottom: $S' = (C'_{c+1}, \ldots, C'_t, C'_1, \ldots, C'_c)$.

The secret parameters of this permutation are called $\boldsymbol{P_c}$ consist of c and all secret mask-parameters. We denote $\boldsymbol{S \overset{P_c}{\sim} S'}$ for this operation. Of course the correctnes of this operation has to be proven. Here, we use the closure of cyclic shifts, i.e. two succeeding cyclic shift operations can be described by one cyclic shift.

Proof of P_c

Step	Alice		All other players
known	Alice has P_c in secret, such that $$S \overset{P_c}{\sim} S' \ .$$		S, S'
1.		$\xleftarrow{s}$	agree on a security parameter s.
2.	Alice chooses s cyclic shift parameters $c_1, \ldots, c_s$ and corresponding $P_{c_1}, \ldots, P_{c_s}$. Using this information Alice creates shifted versions of the stack S', namely $S' \overset{P_{c_1}}{\sim} S_1, \ldots, S' \overset{P_{c_s}}{\sim} S_s$ and publishes them.	$\xrightarrow{S_*}$ $\xleftarrow{X}$	All players agree on a random subset $X \subseteq \{1, \ldots, s\}$.
3.	For all $\ell \in \{1, \ldots, s\}$:		
3a.	If $\ell \in X$ Alice publishes P_{c_ℓ}.	$\xrightarrow{P_{\sigma_\ell}}$	check whether $S' \overset{P_{c_\ell}}{\sim} S_\ell$.
3b.	If $\ell \notin X$: Alice publishes $P_{c_\ell + c}$.	$\xrightarrow{P_{c_\ell + c}}$	check whether $S \overset{P_{c_\ell + c}}{\sim} S_\ell$.

Test of equal cards $\boldsymbol{A \looparrowright B}$ or $\boldsymbol{C \looparrowright D}$ A, B, C, and D are cards where Alice masked $A \looparrowright B$ or $C \looparrowright D$. She wants to show that she performed at least one of these mask-operations.

Proof of $A \looparrowright B$ or $C \looparrowright D$

Step	Alice		All other players
known	mask information of $A \looparrowright B$ or $C \looparrowright D$		reverse sides of A, B, C, D
1.		$\xleftarrow{s}$	agree on a security parameter s.
2.	Alice generates a random subset $X \subseteq \{1, \ldots, s\}$.		
3.	For all $\ell \in \{1, \ldots, s\}$:		
3a.	If $\ell \in X$ then Alice generates $A \looparrowright A'$, $B \looparrowright B'$, $C \looparrowright C'$, $D \looparrowright D'$.	$\xrightarrow{A',B',C',D'}$	
3b.	If $\ell \notin X$ then Alice generates $A \looparrowright C'$, $B \looparrowright D'$, $C \looparrowright A'$, $D \looparrowright B'$.	$\xrightarrow{A',B',C',D'}$	
4.		$\xleftarrow{Y}$	All players agree on a random subset $Y \subseteq \{1, \ldots, s\}$.
5.	For all $\ell \in \{1, \ldots, s\}$:		
5a.	If $\ell \in X \cap Y$ then		
	Alice proves that $A \looparrowright A'$, $B \looparrowright B'$, $C \looparrowright C'$, $D \looparrowright D'$.		
5b.	If $\ell \in Y \setminus X$ then		
	Alice proves that $A \looparrowright C'$, $B \looparrowright D'$, $C \looparrowright A'$, $D \looparrowright B'$.		
5c.	If ($\ell \in X \setminus Y$ and $A \looparrowright B$) or ($\ell \notin X \cup Y$ and $B \looparrowright D$) then		
	Alice proves that $A' \looparrowright B'$		
5d.	If ($\ell \in X \setminus Y$ and $B \looparrowright D$) or ($\ell \notin X \cup Y$ and $A \looparrowright B$) then		
	Alice proves that $C' \looparrowright D'$		

Secretly insert a card into a stack Alice wants to insert a hidden card C into a stack S at a secret position. For this Alice shows that a cyclic shifted version S' of the resulting stack S'' consists of C on the top and a cycled version S in the rest. It remains to prove that both cyclic shifts combined preserve the order of S. Note that the first card of S has to correspond to the first card S'' or the last card S to the last card S''. Alice proves this by the above shown protocol.

Insert a card C in a secret position of a stack S

Step	Alice		All other players
known	reverse sides C, S	C, S	
1.	Alice chooses position $c \in \{1, \ldots, \|S\|\}$ and computes an encoded stack that corresponds to the cyclic shift of S by c cards: $S \sim S'$.	$\xrightarrow{S'}$	
2.	Alice proves that $S \sim S'$		
3.	Alice computes the resulting stack S'' as the cyclic backshifted stack of S' (i.e. shift by $P_{\|S\|-c+1}$) $(C, S') \sim S''$	$\xrightarrow{S''}$	
4.	Alice proves that $(C, S') \sim S''$		
5.	Alice proves that $S[1] \looparrowright S''[1]$ or $S''[\|S\|] \looparrowright S''[\|S\|+1]$		

Glueing and separating cards The encoding of the binary representation gives more opportunities. If partial information like the color of a card is stored in corresponding positions of the binary representation other information can be discarded and further operations may work only with the interesting partial information.

On the other hand, some cards may be glued together, forming a new card data structure which for example may be used to mix up stacks, completely preserving their cards and order. After the mix-operations the separation of the large cards (stacks) gives the original data structure.

Moreover, the binary representation may be used for sharing-of-secrets schemes. Now the pick-up procedure for a player alone does not give all the needed information. The card has to be shared among some players.

Introduction of a new player is a delicate operation, if some game steps already took place, e.g. Charly, the new player, has no guarantee that the others really mixed stacks. Even so, he can testify all protocols despite the verification whether creation of covered random cards and mixing operations were fair against him.

Introduction of a new player

1. Charly publishes and proves his public key.
2. All (private and covered) cards in the game get an additional row containing $(1, \ldots, 1)$.
3. The owners of private cards mask them, all covered cards are masked by Charly.
4. Charly performs secret permutations on all covered stacks that are claimed to be mixed and proves the correctness of each operation.

Leaving a game In some games there are situations where a player, e.g. Charly, is not involved in the rest of the game, e.g. they play "old maid" and a player has laid down all his cards. In the above shown scheme Charly is involved in all the protocols for the rest of the game. If he does not want to control the game anymore, his encodings are removed from the data structure. Of course this operation makes only sense if at least two players continue the game. If only the adversary players' coalition remains, then they can get all information of covered cards which Charly has left for them in the game. Charly's private cards and public covered cards that are not needed for the rest of the game stay securely encoded. Of course the danger stays that some of his strategic decisions or card information may be detected from now on and he is no longer able to keep the game under surveillance. All this has to be considered before using the following protocol:

Parting from a game

Step	Charly		All other players
1.		$\xleftarrow{S}$	ask Charly for the encoding of the covered cards S that remain in the game.
2.	For every card $C \in S$ he publishes his encoding.		
3.	Charly proves his encoding of S.		
4.		$\xleftarrow{S'}$	ask Charly for the proof of his encoding of all their private cards S'.
3.	Charly proves his encoding of S'.		

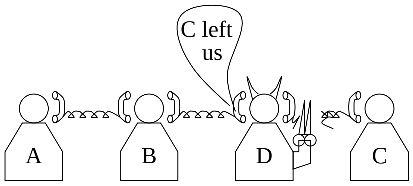

Figure 4: Sabotage! The diabolic player disconnects Charlie and claims that he left without quitting. Alice and Bob cannot disprove that claim.

4 Discussion and Conclusion

4.1 Cheating and Attacks

How can a player cheat or sabotage a game? We classify six categories and discuss whether this toolbox prevents them.

Sabotage This includes all syntactical errors violating the protocol. The most important case is that a player leaves the game without quitting. This case is mostly relevant for a sabotage of a bad loser who wants to spoil the whole game. But note that we cannot differ between a bad loser disconnecting and a local network breakdown. An implementation of an electronic card game should take special care to those kinds of errors (see fig. 4). If a saboteur attacks the game, the only possibility seems to quit the game, identify the saboteur(s) and the kind of attack.

Protocol attack Here the offender tries to attack the cryptographical protocols, e.g. he claims that m has only two prime numbers in its integer factorization, while it has more, he tries to factor m, he uses wrong mask-operations trying to muddle through the proof of it. We have proved that under QRA our scheme is secure against such an attack.

Cheating The offender tries to break the rules of the underlying game, e.g. he serves the wrong card. We have given an example of on-line control of rules that even refer to private cards. Most of the underlying games provide rules for the treatment of cheaters.

Secret coalitions At least two players exchange private information and use it for their advance. At least we can guarantee that their coalition does not enable to turn covered cards or decode private cards of other players. The attempt to try to avoid any secret communication is of course hopeless. There is only one way to ensure that their strategic decisions are independent from the knowledge of an allied partner's private cards: Every player has to fix his strategy before the games starts by programming it into a probabilistic Turing machine. During the game he has to verify that all his (secret) decisions coincide with the (secret) output of this machine. In principle such a scheme can be implemented, but this method fails since most of the human players are not able and willing to describe their strategy.

Public coalitions A coalition of players discriminates a player, only with public known information. A commonly practiced example is a game of three people where two of them are a couple. Here, even the approach of a secret strategic Turing machine does not help, since the strategy could provide sensors that identify the partner by his strategic decisions.

Ghost players A player invents new virtual players to increase the probability of winning. Many games support this strategy, e.g poker, whilst for other games the probability of loosing increases. Like secret and public coalitions ghost players cannot be detected, since the players play anonymously. Since this problem highly depends on the sort of game, players participating these games should always be aware that this can happen.

In the beginning we assumed that all players use a broadcast channel, where everybody hears everything and the sender of a message can be identified. In the real world we have point-to-point-communication which has to emulate the broadcast channel. For identification a digital signature scheme should be used. Signing all messages can prevent also the problem of byzantine agreement when convicting an offender: Consider Charly playing fair with Alice while cheating Bob and claiming that Bob cheats. Now, without signed copies of all messages sent so far, Alice would have no chance to identify Charly as offender.

4.2 Conclusion and Open Problems

We presented an intuitive and simple data structure for cards. Cards may be covered, open, or private to one or more players. Further, we presented protocols for mixing a stack consisting of open and hidden cards, picking up cards, opening cards, inserting cards in the stack, splitting the stack, and many more. It is possible to control that players obey the rules even when they make hidden moves without disclosure of their secrets.
All proofs can be repeated such that it is always possible to decrease the probability that a protocol attack was not detected. Only the insecurity caused by too small secret keys cannot be diminished. Like in Crépeau's scheme private cards have not to be revealed after the game keeping the player's strategy secret. Furthermore, secret and public keys can be used for many games.
At last we discussed the problem of secret and public coalitions and ghost players in mental card games. It is an open problem how these problems can be solved in practice. Another open problem is the question how rules can be controlled just in time applying to covered cards without publishing this secret information. Further, it is open whether and how other cryptographical methods, e.g. discrete logarithm, or modular exponentiation, can replace the quadratic residuosity used here.

Acknowledgement

This work was inspired by the discussions with a group of interested students, namely Thomas Arand, Matthias Holm, Matthias Kolberg. Further I have to thank Andreas Jakoby for the very detailed explanation of mixing tradition in "Doppelkopf" which led me to the glueing and separation operation. Special thanks are given to Stephan Weis and Barbara Goedecke for proof-reading and Rüdiger Reischuk for drawing my attention to cryptographic protocols.
At last I want to mention that there is at our institut an ongoing student project of an interactive, graphical implementation of parts of this system that in its final state will allow a transparant secure mental card game in the internet.

References

[BaFü 83] I. Banary, Z. Füredi, *Mental Poker with Three or More Players*, Information and Control, 59, pp. 84-93, 1983.

[BrCr 86] G. Brassard, C. Crépeau, *Zero-Knowledge Simulation of Boolean Circuits*, Crypto 86, LNCS 263, pp. 223-233, 1986.

[BCR 87] G. Brassard, C. Crépeau, J.-M. Robert, *All-or-nothing disclosure of secrets (extended abstract)*, Crypto'86, pp. 234-238, 1987.

[Blum 81] M. Blum, *Coin Flipping by Telephone, A Protocol for Solving Impossible Problems*, SIGACT News, 1981, pp. 23-27, 1987.

[Crép 86] C. Crépeau, *A Secure Poker Protocol That Minimizes the Effect of Player Coalitions*, Adavances in Cryptology: Proc. of Crypto 85, LNCS 218, Springer, pp. 73-86, 1986

[Crép 87] C. Crépeau, *A Zero-Knowledge Poker Protocol that Achieves Confidentiality of the Players' Strategy or How to Achieve an Electronic Poker Face*, Crypto'86, pp. 239-247, 1987.

[FoMe 85] S. Fortune, M. Merrit, *Poker Protocols*, Advances in Cryptology: Proc. of Crypto 84, LNCS 196, Springer, pp. 454-464, 1985.

[GHY 85] Z. Galil, S. Haber, M. Yung, *A Private Interactive Test of a Boolean Predicate and Minimum-Knowledge Public-Key Cryptosystems*, FoCS'85, pp. 360-371.

[GoMi 82] S. Goldwasser, S. Micali, *Probabilistic Encryption & How To Play Mental Poker Keeping Secret All Partial Information*, SToC'82, pp. 365-377, 1982.

[GMR 85] S. Goldwasser, S. Micali, C. Rackoff, *The knowledge complexity of interactive proof-systems*, SToC'85, pp. 281-304, 1985.

[GMW 87] O. Goldreich, S. Micali, A. Widgerson, *How to Play any Mental Game or a Completeness Theorem for Protocols with Honest Majority*, SToC'87, pp. 218-229, 1987.

[Jako 98] A. Jakoby, *Private Communication*, 1998.

[SRA 78] A. Shamir, R. Rivest, L. Adleman, *Mental Poker*, MIT Technical Report, 1978.

[Yung 85] M. Yung, *Cryptoprotocols: Subscription to a Public Key, The Secret Blocking and the Multi-Player Mental Poker Game*, Advances in Cryptology: Proc. of Crypto 84, LNCS 196, Spring, pp. 439-453, 1985.