# Mental Poker Revisited

Adam Barnett and Nigel P. Smart

Department of Computer Science,
University of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB,
United Kingdom.
{ab9660,nigel}@cs.bris.ac.uk

**Abstract.** We discuss how to implement a secure card game without the need for a trusted dealer, a problem often denoted "Mental Poker" in the literature. Our solution requires a broadcast channel between all players and the number of bits needed to represent each card is independent of the number of players. Traditional solutions to "Mental Poker" require a linear relation between the number of players and the number of bits required to represent each card.

## 1 Introduction

The notion of playing card games "via telephone", or "Mental Poker" as it is sometimes called, has been historically important in the development of cryptography. Mental Poker protocols enable a group of mutually mistrusting players to play cards electronically without the need for trusted dealers. Such a set of protocols should allow the normal card operations (e.g. shuffling, dealing, hiding of card information from players) to be conducted in a way which allows cheaters to be detected.

The original scheme for this problem was developed by Shamir, Rivest and Adleman [15] not long after the publication of the RSA algorithm itself. However, this was soon noticed to be insecure since the naive RSA function leaks information about individual bits. This observation led Goldwasser and Micali [10,11] to develop the notion of semantic security and the need for probabilistic public key encryption. Thus Mental Poker led to the current subject of provable security and the current definitions of what it means for a public key encryption scheme to be secure.

After the work of RSA and Goldwasser and Micali there has been other work on how to conduct two-player card games over a public network with no trusted centre, see [1,6,7,9,12,17]. A full protocol suite for an arbitrary card game with an arbitrary number of players and with no trusted centre is described by Schindelhauer [16]. However, all of these schemes make use of the Goldwasser–Micali probabilistic encryption scheme [11] based on the quadratic residuosity problem, but the Goldwasser-Micali system is very inefficient since it requires a full RSA-style block to encrypt a single bit of information.

K.G. Paterson (Ed.): Cryptography and Coding 2003, LNCS 2898, pp. 370–383, 2003.

As a result the above schemes require

$$l \times \lceil \log_2 M \rceil \times k$$

bits to represent each card where

- $l$ is the number of players in the game.
- $M$ is the number of different cards, usually $M = 52$.
- $k$ is the number of bits in a secure RSA modulus.

Our protocols will give message sizes which are independent of the number of players. In addition our protocol is essentially independent of the number of different cards in the game, at least for any game which one could imagine.

We give two instantiations of our scheme, one based on discrete logarithms and one based on a factoring assumption (actually Paillier's system [13]). We therefore require only $k$ bits to represent each card, where using current security recommendations,

- $k = 322$ if a 160-bit elliptic curve is chosen for the discrete logarithm based variant, and point compression is used.
- $k = 2048$ if a discrete logarithm based variant is used in a finite field of order $\approx 2^{1024}$.
- $k = 2048$ if the Paillier variant is used with an RSA modulus of 1024 bits.

However unlike the system described in [16] for our factoring based protocol we assume that players may not join an existing game, however they may leave. For both discrete logarithm based versions players may both leave and join a game as it proceeds.

The paper is structured as follows. Firstly we define some notions related to proofs of knowledge which we shall require. Then we introduce the notion of a Verifiable $l$-out-of-$l$ Threshold Masking Function, or VTMF, which is the basic cryptographic operation which we will apply repeatedly to our cards. We then go on to show how the protocols to implement the card game are implemented via a VTMF. Then we give descriptions of two VTMF's, one based on discrete logarithms and one based on Paillier's assumption [13].

We end this introduction by noting that in some sense Mental Poker makes no sense in a model which allows collusion. For example one could use the following protocol suite to create an electronic Bridge game. However, two partners, say East and West, could contact each other by some other means, i.e. not using the broadcast channel, and exchange information which would enable them to have an advantage over North and South. No Mental Poker suite of protocols could detect such cheating. However, the protocols which follow ensure that the colluding parties obtain no more information than if they had colluded in a real game.

## 2 Proofs of Knowledge

In the following sections we will make reference to various properties of proofs of knowledge. Much of this introductory section summarizes the work of Cramer, Damgård and Schoenmakers [5].

Consider a binary relation $R = \{(x, w)\}$ for which membership can be tested in polynomial time. For any $x$ we say that $w(x)$ is the set of witnesses such that $(x, w) \in R$. A proof of knowledge protocol $\mathcal{P}$ is a two party protocol between a prover $P$ and a verifier $V$. The prover and verifier are given a common input $x$ and the prover has a private input $w$. The prover's aim is to convince the verifier that $w \in w(x)$ without revealing what $w$ actually is.

Following Cramer et. al. we restrict to the following subset of such protocols. We assume that the protocol is a three round public coin protocol in that the protocol is an ordered triple

$$m_1, c, m_2$$

where $m_1$ is called the commitment and comes from the prover, $c$ is a random challenge chosen by the verifier and $m_2$ is the provers final response.

We assume the following three properties:

1. $\mathcal{P}$ is complete.
   If $w \in w(x)$ then the verifier will accept with probability one.
2. $\mathcal{P}$ has the "special soundness" property.
   For any prover $\mathcal{P}$ given two conversations between $P$ and $V$ with message flows
   $$(m_1, c, m_2) \text{ and } (m_1, c', m_2')$$
   with $c \neq c'$ then an element of $w(x)$ can be computed in polynomial time.
3. $\mathcal{P}$ is honest verifier zero-knowledge.
   There is a simulation $\mathcal{S}$ of $\mathcal{P}$ that on input $x$ produces triples which are indistinguishable from genuine triples between an honest prover and an honest verifier.

To fix ideas consider the Schnorr identification scheme which identifies users on proof of knowledge of a discrete logarithm $h = g^x$. We have

$$m_1 = g^k, m_2 = k + x \cdot c$$

for some random $k$ chosen by $P$ and some random $c$, chosen by $V$ after $P$ has published $m_1$. The verifier checks that

$$m_1 = g^{m_2} \cdot h^{-c}.$$

This satisfies all the properties above. Indeed it is the "special soundness" property which allows Pointcheval and Stern [14] to show that the associated signature scheme derived via the Fiat–Shamir paradigm is secure.

Cramer et. al. use the above three round honest verifier proofs of knowledge with the special soundness property to create proofs of knowledge of elements

for arbitrary access structures. For example if $P$ the discrete logarithm $x_i$ of one of $h_1 = g^{x_1}, h_2 = g^{x_2}, h_3 = g^{x_3}$. Using the protocols of [5], $P$ can convince $V$ that they know one of the discrete logarithms without revealing which one.

We shall use such protocols to allow players to show that an encryption of a card is an encryption of a card from a given set, or that an encrypted card comes from the set of hearts, or that a player has no hearts left in their hand, etc. etc.

## 3 Verifiable $l$-out-of-$l$ Threshold Masking Functions

For our later protocols we will require a set of cryptographic protocols, which we call a Verifiable $l$-out-of-$l$ Threshold Masking Function, or VTMF for short. In a later section we shall give two examples of such a function, one based on a discrete logarithm assumption and one based on a factoring assumption.

In keeping with the notation in the rest of the paper we shall assume there are $l$ players and there are $M$ values which are to be encrypted (or masked). A VTMF is a set of protocols, to be described in detail below, which produces a semantically secure encryption function (under passive adversaries) from a space $\mathcal{M}$ to a space $\mathcal{C}$. We shall assume that there is a natural encoding

$$\{1, \ldots, M\} \longrightarrow \mathcal{M}$$

which allows us to refer to messages and card values as the same thing. In addition there is a nonce-space $\mathcal{R}$ which is sufficiently large and from which nonces are chosen uniformly at random.

A VTMF consists of the following four protocols:

### 3.1 Key Generation Protocol

This is a multi-party protocol between the $l$ parties which generates a single public key $h$. Each player will also generate a secret $x_i$ and a public commitment to this share, denoted $h_i$. The shares $x_i$ are shares of the unknown private key, $x$, corresponding to $h$.

### 3.2 Verifiable Masking Protocol

A masking function is an encryption function

$$\mathcal{E}_h : \begin{cases} \mathcal{M} \times \mathcal{R} \longrightarrow & \mathcal{C} \\ (m, r) \longmapsto & \mathcal{E}_h(m; r), \end{cases}$$

and an associated decryption function

$$\mathcal{D}_h : \begin{cases} \mathcal{C} & \longrightarrow \mathcal{M} \\ \mathcal{E}_h(m; r) & \longmapsto m, \end{cases}$$

with respect to the public key $h$, we abuse notation and equate knowledge of the private key $x$ with knowledge of the function $\mathcal{D}_h$. In addition there is a honest-verifier zero-knowledge protocol to allow the masking player to verify to any other player that the given ciphertext $\mathcal{E}_h(m;r)$ is an encryption of the message $m$. Since we will use this protocol in a non-interactive manner using the Fiat-Shamir transform the fact it is only honest-verifier will be no problem, as we will achieve security in the random oracle model. We insist that the encryption function is semantically secure under passive attacks. However, we cannot achieve semantic security under active attacks since we also require the ability to re-mask a message, as we shall now explain.

### 3.3 Verifiable Re-masking Protocol

Let $\mathcal{C}_m$ denote the set of all possible encryptions of a message $m$ under the masking function above. Given an element $c$ of $\mathcal{C}_m$ there is a function which re-encrypts $c$ to give another representative in $\mathcal{C}_m$, and which can be applied without knowledge of the underlying plaintext $m$ and only knowledge of the public key $h$. Hence, this function can be applied either by the player who originally masked the card or by any other player. We shall denote this function by $\mathcal{E}'$,

$$\mathcal{E}'_h : \begin{cases} \mathcal{C}_m \times \mathcal{R} \longrightarrow & \mathcal{C}_m \\ (c, r) \longmapsto & \mathcal{E}'_h(c;r). \end{cases}$$

Again we also insist that there is a honest-verifier zero-knowledge protocol to allow the player conducting the re-masking to verify to any other player that the given ciphertext $\mathcal{E}'_h(c;r)$ is an encryption of the message $m$, without either player needing to know the underlying plaintext message $m$.

We insist that if $r \in \mathcal{R}$ is chosen uniformly at random then $\mathcal{E}'_h(c;r)$ is also uniformly distributed over $\mathcal{C}_m$. In addition we also insist that if a user knows $r_1$ and $r_2$ such that

$$c_1 = \mathcal{E}'_h(c;r_1) \text{ and } c_2 = \mathcal{E}'_h(c_1, r_2)$$

then they can compute $r$ such that

$$c_2 = \mathcal{E}'_h(c;r).$$

This last property is needed so that certain proofs of knowledge can be executed.

### 3.4 Verifiable Decryption Protocol

Given a ciphertext $c \in \mathcal{C}$ this is a protocol in which each player generates a value $d_i = \mathcal{D}(c, x_i)$ and an honest-verifier zero-knowledge proof that the value $d_i$ is consistent with both $c$ and the original commitment $h_i$.

We assume there is a public function

$$\mathcal{D}'(c, d_1, \ldots, d_l) = m$$

which decrypts the ciphertext $c$ given the values $d_1, \ldots, d_l$. This function should be an $l$-out-of-$l$ threshold scheme in that no subset of $\{d_1, \ldots, d_l\}$ should be able to determine any partial information about $m$.

# 4 The Card Protocols

For the purpose of this paper, we assume an authentic broadcast channel between all parties. This can be achieved easily in practice using digital signatures and a passive bulletin board to hold the player communications. There are two types of operations; operations on a single card and operations on a deck (or set) of cards.

## 4.1 Operations on a Card

We assume that the players have executed the key set up phase for an $l$-out-of-$l$ VTMF as above, where $l$ is the number of players. Following the approach of [16], we describe the card operations needed in most games. Each card will be represented by an element $c$ of the space $\mathcal{C}$. The value of the card is the unique $m \in \mathcal{M}$ such that $c = \mathcal{E}_h(m; r)$ for some value $r \in \mathcal{R}$. A card is called *open* if $c = \mathcal{E}_h(m; r)$, and $r$ is a publically known value.

**Creation of an open card.** The creation of an open card requires only the input of one player. To create a card with type $m$, the player simply creates card value $c = \mathcal{E}_h(m; 1)$. This format can be read and understood by everyone, so verification of the precise card value is trivial.

**Masking a card.** Masking a card is the application of a cryptographic function that hides the value or type of the card. The homomorphic property of the encryption scheme allows an already masked card to be re-masked. Also, a zero knowledge proof exists that allows the verifier to show that the masked card is the mask of the original card. Thus masking is achieved by use of the verifiable re-masking function $\mathcal{E}'_h$ described above.

**Creation of a private card.** To create a private card, Alice privately creates a card $m$ and then masks it to give $c = \mathcal{E}_h(m; r)$ which she then broadcasts to all other players. The purpose of broadcasting the masked card is to commit to the generated card and to prevent Alice from generating a wild card. The proof of knowledge here has to be slightly different to that described above since we need to show that the card is a mask of a valid card, i.e. $m$ is a genuine card value. This proof is accomplished by the protocol of Cramer et. al. [5] in which one can show that $c$ is the masking of an element of the required subset $\mathcal{M}$.

**Creation of a random covered card.** Unlike the scheme proposed by Schindelhauer [16], there is no trivial way of generating a random covered card unless the number of cards is equal to the number of plaintexts of the underlying encryption scheme. In our instantiations below the size of the underlying plaintext space is exponential and so this approach is not going to work.

There are two possible solutions to this problem:

- One option is to enable the entire underlying plaintext space $\mathcal{P}$ to be considered as card values. Hence, one would require a cryptographic hash function

$$H : \mathcal{P} \longrightarrow \mathcal{M}$$

  which maps plaintexts to possible card values. This assumes that every ciphertext corresponds to a valid encryption, which in our instantiations is a valid assumption.
- The second solution is to create a deck containing all the possible cards that could be randomly generated. Each player then shuffles and masks the deck and a card is chosen at random. This is a more costly method, but distributes the probability of selection correctly while giving no player a chance to influence the generation of the card. Operations on a desk, such as shuffling, are considered later.

**Opening a card.** To open a card, each player executes the verifiable decryption protocol. In the case of publicly opening a card, this information is broadcast to everyone. In the case of privately opening a card one player keeps their decryption information secret. Thus enabling them to read the card. The associated proofs in the verifiable decryption protocol are used to ensure that no player can sabotage the game by providing incorrect decryptions.

There are other possible card opening situations such as when a card needs to be opened by two players, $N$ and $S$, but no others. Due to the broadcast nature of our network we need to solve this using a two stage process: One player, $N$ say, first re-masks the card and provides a proof of the re-masking. The group minus $N$ and $S$, then contribute their shares of the decryption of both cards. Player $N$ contributes a decryption of the card to be opened by $S$, and $S$ contributes a decryption of the card to be opened by $N$. Similar situations can be dealt with by adapting the above ideas.

### 4.2 Operations on a Deck

A deck $D$ is modelled as a stack of cards of size $n$. There is no reason for there to be only one deck in the game, and each players hand could be considered a deck in itself. At the start of the game a player may create the deck. The notation for masking a card $\mathcal{E}_h(m; r)$ is often abused to also mean masking the entire deck in the form $\mathcal{E}_h(D; R)$, where $R = \{r_1 \dots r_n\}$. In this case, each card is masked individually using the corresponding $r$ from $R$.

**Creating the deck.** This operation is equivalent to the creation of several open cards which are then stacked and mask-shuffled. In order to ensure that the player who created the deck does not follow where the cards are, each player must mask-shuffle the deck before play can begin. Any player who does not mask-shuffle the deck could have all of their cards discovered by the collusion of his opponents.

**Mask-shuffling the deck.** We define a composite operation called mask-shuffle which both shuffles the deck and applies a mask to each individual card in the deck. To mask-shuffle the deck, the player must apply a permutation and then mask each card. If parts of the deck have not yet been masked (either due to the deck just being created or the player adding a card to the deck), the player knows what card has been masked to what value, i.e. it is *privately masked*. Therefore, it is often the case that the deck must be re-masked by all other players to ensure it is *publicly masked*.

Also, the mask-shuffle operation includes a verification stage to ensure the mask-shuffle has been performed correctly, i.e. the resulting deck is semantically the same as the initial deck. This is done using an honest verifier zero-knowledge proof, as in Figure 1.

For the purposes of this proof, it is necessary to keep track of all the permutations and $R$s used during the initial masking of $D$ to form $D'$. However, once the mask-shuffle has been verified as correct they may be discarded. Note that the chance of cheating has been reduced to $\frac{1}{2^s}$, so $s$ should be suitably large to satisfy all parties.

**Splitting the deck.** Splitting a deck means that a player moves $x$ cards from the top of the deck to the bottom, maintaining the order of the removed cards. This is a similar operation to shuffling the deck, in that a permutation can be used to represent the split. However, the proof needs to be extended in order to show that a split has occurred and *not* a regular shuffle. The proof is similar to the honest verifier zero knowledge proof used for shuffling the deck, on noticing that a split followed by a split is simply another split. Figure 2 shows the protocol in more detail.

**Drawing a card from the deck.** This operation is equivalent to privately opening a masked card. In this case, the player broadcasts to all players which card he is going to draw. The card is then privately opened once all players have agreed to share enough information to do so. In some instances a player might be restricted to drawing only from the top of the stack. As each player has a copy of the stack this would be possible. Note that if a player has just placed a card on the stack, the player who takes this card will be letting the other player know what card is in his hand. This is acceptable as it would be true in a real game of cards. However, the player who draws this card should re-mask his hand if he ever intends to discard a card. This prevents the other player knowing when he has discarded the card given to him. Discarding cards is considered later.

**Discarding a card from hand.** A players hand is made up of a series of masked cards which they know the decryption of. However, there are some times when an opponent knows what card you are holding at a certain point (for example, passing cards in Hearts). In this case, the player must always mask-shuffle his hand whenever he intends to discard a card from it. For example, if a player wants to insert a card from his hand into the deck, he must first *discard* this card and then insert it into the deck. This operation ensures that no information about a players hand is revealed.

| Alice | Bob |
|---|---|
| A permutation $\sigma$ is applied to the deck $D$, and each element is masked to create $D'$ such that $$D' = \mathcal{E}_h(\sigma(D); R).$$ Alice publishes the mask-shuffled deck $D'$ | |
| | Bob chooses a security parameter $s$ and sends this to Alice. |
| Alice generates a set of permutations $\{\sigma'_1, \ldots, \sigma'_s\}$ and sends $D''_i$ to Bob where $$D''_i = \mathcal{E}_h(\sigma'_i(D'); R_i).$$ | |
| | Bob chooses a random subset of $X \subset \{D''_1, \ldots, D''_s\}$ and sends this to Alice. |
| For all $D''_i \in X$, Alice publishes $$\sigma'_i, R_i$$ otherwise she publishes $$\sigma \circ \sigma'_i, R'_i$$ where $R'_i$ are the masking values used to mask $D''_i$ from $D$. These later values should be easily computable from $R$ and $R_i$ | Bob verifies that Alice has supplied the correct masking values and permutations to go from $D'$ to $D''_i$ if $D''_i \in X$ or from $D$ to $D''_i$ if $D''_i \notin X$. |

**Fig. 1.** Mask-Shuffling the Deck

**Rule control.** The use of these set tests allow for rule control that are not available in real card games. For example, in whilst style games such as Hearts and Bridge, a player must "Follow Suit" whenever possible. A cheating player could choose to not follow suit for his own advantage. Although the player would discovered to be cheating at the end of the game, it would be preferable to discover the player to be cheating earlier.

By using the protocol of Cramer, Damgård and Schoenmakers [5] a player can always prove when playing a card that it comes from a given set. Hence, one can make sure that players follow rules, such as following suit, as the game progresses.

| **Alice** | **Bob** |
|---|---|
| A split $S$ is applied to the deck $D$, and each element is masked to create $D'$ such that<br>$$D' = \mathcal{E}_h(S(D); R).$$ | Bob chooses a security parameter $s$ and sends this to Alice. |
| Alice generates a further set of masked splits $\{S'_1, \ldots, S'_s\}$ and sends $D''_i$ to Bob where<br>$$D''_i = \mathcal{E}_h(S'_i(D'); R_i).$$ | |
| | Bob chooses a random subset of $X \subset \{D''_1, \ldots, D''_s\}$ and sends this to Alice. |
| For all $D''_i \in X$, Alice publishes<br>$$S'_i, R_i$$<br>otherwise she publishes<br>$$S \circ S'_i, R'_i$$<br>where $R'_i$ are the masking values used to mask $D''_i$ from $D$. These later values should be easily computable from $R$ and $R_i$ | Bob verifies that Alice has supplied the correct masking values and permutations to go from $D'$ to $D''_i$ if $D''_i \in X$ or from $D$ to $D''_i$ if $D''_i \notin X$. |

**Fig. 2.** Splitting the Deck

**Leaving the game.** Although new players are unable to enter the game, players are able to leave the game whenever they wish. In order to do so, any cards that the player has in their hand must either be returned to the deck or opened and discarded (depending on the rules of the game). Then, each remaining player mask-shuffles the deck and their hands. Once this has been done, the departing player reveals their secret key and verifies it to be correct. This allows all the remaining cards to still be decrypted despite a player not being present. Note that it is not generally possible for the player to re-enter the game once they have left, however for our discrete logarithm based VTMF below we shall show that this is possible.

## 5 Instantiations of VTMF's

We now present two examples of a VTMF suitable for use in our card protocols. The first is based on the ElGamal encryption scheme whilst the second is based on Paillier's system [13]. Before presenting the instantiations we require the following sub-protocols, both of which are honest-verifier zero-knowledge and possess the "special soundness" property of [5].

**Proof of Knowledge of Equality of Discrete Logarithms**
The following protocol, due to Chaum and Pedersen [4], provides a proof, that if the verifier is given $x = g^\alpha$ and $y = h^\beta$ then the prover knows $\alpha$ and that $\alpha = \beta$, where $g$ and $h$ have order $q$.

- The prover sends the commitment $(a, b) = (g^\omega, h^\omega)$ to the verifier, for some random value $\omega \in \mathbb{Z}_q$.
- The verifier sends back a random challenge $c \in \mathbb{Z}_q$.
- The prover responds with $r = \omega + \alpha c \pmod q$.
- The verifier accepts the proof if and only if he verifies that $g^r = ax^c$ and $h^r = by^c$.

We shall denote this protocol by $CP(x, y, g, h; \alpha)$.

**Proof of $n^{th}$ residuosity modulo $n^2$**
The following protocol, due to Damgård and Jurik [8], provides a proof that a value $u \in \mathbb{Z}_{n^2}$ is a perfect $n^{th}$ power and that the prover knows an $n^{th}$ root $v$.

- The prover sends the commitment $a = r^n \pmod{n^2}$ for some random value $r \in \mathbb{Z}_{n^2}$.
- The verifier sends back a random challenge $c \in \mathbb{Z}_n$.
- The prover responds with $z = r \cdot v^c \pmod{n^2}$.
- The verifier accepts the proof if and only if $z^n = a \cdot u^c \pmod{n^2}$.

We shall denote this protocol by $DJ(n, u; v)$.

When we apply these protocols the random challenge $c$ will be created from hashing the commitment and the public input values, thus making the protocol non-interactive. In such a situation we also denote the transcript of the proofs by $CP(x, y, g, h; \alpha)$ and $DJ(n, u; v)$.

### 5.1 Discrete Logarithm Based VTMF

The parties first agree on a finite abelian group $G$ in which the Decision Diffie–Hellman assumption is hard. The users agree on a generate $g \in G$ of order $q$ and set

$$\mathcal{M} = G,\ \mathcal{R} = \mathbb{Z}_q \text{ and } \mathcal{C} = G \times G.$$

**Key Generation Protocol**
Each player generates a random private key $x_i \in \mathbb{Z}_q$ and publishes $h_i = g^{x_i}$. The public key $h$ is formed from

$$h = \prod_{i=1}^{l} h_i.$$

**Verifiable Masking Protocol**
The verifiable masking protocol is given by the ElGamal encryption operation

$$(m, r) \longrightarrow (c_1 = g^r, c_2 = h^r \cdot m).$$

The value $(c_1, c_2)$ is published and is accompanied by the proof

$$CP(c_1, c_2/m, g, h; r).$$

That this encryption function is semantically secure under the Decision Diffie-Hellman assumption is an undergraduate exercise.

**Verifiable Re-masking Protocol**
Given a ciphertext $(c_1, c_2)$ this is re-masked by computing

$$((c_1, c_2), r)) \longrightarrow (c_1' = c_1 \cdot g^r, c_2' = c_2 \cdot h_r).$$

The value $(c_1', c_2')$ is now published and accompanied by the proof

$$CP(c_1'/c_1, c_2'/c_2, g, h; r).$$

**Verifiable Decryption Protocol**
Given $(c_1, c_2)$, user $i$ publishes $d_i = c_1^{x_i}$ along with a proof $CP(d_i, h_i, c_1, g; x_i)$. Given these values any player can decrypt $(c_1, c_2)$ by computing

$$m = c_2 / \prod_{i=1}^{l} d_i.$$

**Joining The Game**
Using this VTMF a player may join an existing card game by generating a private key $x_{l+1}$ and public commitment $h_{l+1}$ as above. The public key $h$ for the game is then replaced by

$$h' = h \cdot h_{l+1}.$$

Then each card $c = (c_1, c_2)$ is masked by the new player, under the new public key, by setting

$$c' = (c_1', c_2') = (c_1, c_1^{x_{l+1}} \cdot c_2)$$

along with a proof $CP(h_{l+1}, c_2'/c_2, g, c_1; x_{l+1})$.

### 5.2 Factoring Based VTMF

Our factoring based assumption will make use of the Paillier probabilistic encryption function [13], which is known to be semantically secure against passive attacks under the $n$-th residuosity assumption.

**Key Generation Protocol**
All parties execute a protocol such as that of Boneh and Franklin, see [2,3], to generate a shared RSA modulus $n = p \cdot q$, where each player only knows the value of $(p_i, q_i)$ where $p = \sum_{i=1}^{l} p_i$ and $q = \sum_{i=1}^{l} q_i$. The value $n$ is published and the users generate a share of $\phi = (p-1)(q-1)$ by setting

$$x_i = \begin{cases} n - (p_1 + q_1) - 1 & \text{If } i = 1, \\ -(p_i + q_i) & \text{If } i \geq 2. \end{cases}$$

Note that $\phi = \sum_{i=1}^{l} x_i$. The uses then commit to the value $x_i$ by publishing $h_i = g^{x_i} \pmod{n^2}$, where $g = 1 + n$. We then set publically

$$h = \prod_{i=1}^{m} h_i - 1 \pmod{n^2} = g^{\phi} - 1 \pmod{n^2}.$$

**Verifiable Masking Protocol**
The verifiable masking protocol is given by Paillier's encryption operation

$$(m, r) \longrightarrow c = g^m r^n \pmod{n^2}.$$

The value $c$ is published and is accompanied by the proof $DJ(n, c/g^m; r)$.

**Verifiable Re-masking Protocol**
Given a ciphertext $c$ this is re-masked by computing

$$(c, r) \longrightarrow c' = r^n c \pmod{n^2}.$$

The value $c'$ is now published and accompanied by the proof $DJ(n, c'/c; r)$.

**Verifiable Decryption Protocol**
Given $c$, user $i$ publishes the value $d_i = c^{x_i} \pmod{n^2}$ along with a proof $CP(d_i, h_i, c, g; x_i)$. Given these values any player can decrypt $c$ by computing

$$\begin{aligned} \frac{1}{h}\left(\left(\prod_{i=1}^{l} d_i\right) - 1 \pmod{n^2}\right) \pmod{n} &= \frac{c^{\phi} - 1 \pmod{n^2}}{g^{\phi} - 1 \pmod{n^2}} \pmod{n} \\ &= m. \end{aligned}$$

## 6 Conclusion

We have introduced the concept of a VTMF and shown how this can be used to implement a secure multi-party card game with arbitrary number of players and

no trusted centre. The number of bits to represent each card in our system is significantly smaller than in previous schemes. We have then gone on to show how a VTMF can be implemented either under a discrete logarithm type assumption or under a factoring based assumption.

The authors would like to thank the referee's for very useful comments which improved the readability of the paper. The authors would also like to thank F. Vercauteren for useful conversations whilst the work in this paper was carried out.

## References

1. I. Banary and Z. Füredi. Mental poker with three or more players. *Information and Control*, **59**, 84–93, 1983.
2. D. Boneh and M. Franklin. Efficient generation of shared RSA keys. *Advances in Cryptology – CRYPTO '97*, Springer-Verlag LNCS 1233, 425–439, 1997.
3. D. Boneh and M. Franklin. Efficient generation of shared RSA keys. **J. ACM**, **48**, 702–722, 2001.
4. D. Chaum and T.P. Pedersen. Wallet databases with observers. *Advances in Cryptology – CRYPTO '92*, Springer-Verlag LNCS 740, 89–105, 1993.
5. R. Cramer, I. Damgård and B. Schoenmakers. Proofs of partial knowledge. *Advances in Cryptology – CRYPTO '94*, Springer-Verlag LNCS 839, 174–187, 1994.
6. C. Crépeau. A secure poker protocol that minimises the effect of player coalitions. *Advances in Cryptology – CRYPTO '85*, Springer-Verlag LNCS 218, 73–86, 1986.
7. C. Crépeau. A zero-knowledge poker protocol that achieves confidentiality of the player's strategy, or how to achieve an electronic poker face. *Advances in Cryptology – CRYPTO '86*, Springer-Verlag LNCS 263, 239–247, 1987.
8. I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. *Public Key Cryptography – PKC 2001*, Springer-Verlag LNCS 1992, 119–136, 2001.
9. S. Fortune and M. Merrit. Poker protocols. *Advances in Cryptology – CRYPTO '84*, Springer-Verlag LNCS 196, 454–464, 1985.
10. S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. *STOC '82*, 365–377, 1982.
11. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and Systems Sciences*, **28**, 270–299, 1984.
12. O. Goldreich, S. Micali and A. Widgerson. How to play any mental game or a completeness theorem for protocols with honest majority. *STOC '87*, 218–229, 1987.
13. P. Paillier. Public key cryptosystems based on composite residue classes. *Advances in Cryptology – EuroCrypt '99*, Springer-Verlag LNCS 1592, 223–238, 1999.
14. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, **13**, 361–396, 2000.
15. A. Shamir, R. Rivest and L. Adleman. Mental Poker. *MIT Technical Report*, 1978.
16. C. Schindelhauer. A toolbox for mental card games. *Technical Report*, Uni Lubeck, 1998.
17. M. Yung. Subscription to a public key, the secret blocking and the multi-party poker game. *Advances in Cryptology – CRYPTO '84*, Springer-Verlag LNCS 196, 439–453, 1985.