Practical Mental Poker Without a TTP Based on Homomorphic Encryption

Jordi Castellà-Roca[1], Josep Domingo-Ferrer[2], Andreu Riera[1], and Joan Borrell[3]

[1] Scytl Online World Security S.A., Entença 95 4-1, E-08015 Barcelona, Catalonia.
{jordi.castella,andreu.riera}@scytl.com
http://www.scytl.com
[2] Universitat Rovira i Virgili, Dept. of Computer Engineering and Maths,
Av. Països Catalans 26, E-43007 Tarragona, Catalonia.
jdomingo@etse.urv.es
[3] Universitat Autònoma de Barcelona, Dept. of Computer Science,
E-08193 Bellaterra, Catalonia.
joan.borrell@uab.es

Abstract. A solution for obtaining impartial random values in on-line gambling is presented in this paper. Unlike most previous proposals, our method does not require any TTP and allows e-gambling to reach standards of fairness, security an auditability similar to those common in physical gambling.
Although our solution is detailed here for the particular case of games with reversed cards (*e.g.* poker), it can be easily adapted for games with open cards (*e.g.* blackjack) and for random draw games (*e.g.* keno). Thanks to the use of permutations of homomorphically encrypted cards, the protocols described have moderate computational requirements.
Keywords: Mental poker, E-gambling, Privacy homomorphisms.
Categories: Applications of cryptography (e-gambling), Multi-party computation.

1 Introduction

Computer networks and especially the Internet have allowed some common activities such as shopping, information search o gambling to become remote. This paper is about gambling over the Internet, also called e-gambling, and more specifically about mental poker or e-poker. E-gambling has a number of advantages for players, because it is space-independent (there is no need to physically go to the casino) and time-independent (an Internet casino can be available 24 hours a day or at least longer than physical casinos). The drawback of e-gambling is the difficulty of guaranteeing the same standards of security, fairness and auditability offered by physical gambling.

In a physical casino, each player sees the actions performed by other players, so that most unfair actions can immediately be detected and reported. In the digital world, things are more complex, as discussed below:

T. Johansson and S. Maitra (Eds.): INDOCRYPT 2003, LNCS 2904, pp. 280–294, 2003.

Card draw In a physical casino, any player can draw a card without the other players seeing the card she gets. In an Internet casino, however, unless some kind of encryption is used, a third party can see the card gotten by any player.

Randomness In a physical casino, physical devices can be used to obtain true randomness (roulettes, card shuffling, dice, etc.). The fairness, the unbiasedness and the outcomes of such devices can be verified by all players (at least in theory). In an on-line casino, the best that can be achieved is pseudo-randomness and verifying fairness is much less trivial. Pseudo-random values should be generated in a way that they cannot be manipulated. A common way to achieve this goal is to use a Trusted Third Party (TTP). The usual setting is for the on-line casino to act as a TTP while, at the same time, playing an active role in the game. In this case, the on-line casino is in a privileged position, whereas players are helpless and cannot verify actions by other players or by the casino. As reported in [13], unlawful manipulation in remote gambling is quite common. A better option is for random values to be jointly generated by all participants using a cryptographic protocol. This is the approach proposed in this paper; our cryptographic protocols guarantee that, if the random value has been manipulated by some participant, such manipulation is detected.

Auditability Physical casinos are equipped with CCTV or other recording systems which allow dispute resolution when there is disagreement between players and the casino or between players themselves as to the development or outcome of the game. Auditing misbehaviors in on-line casinos is less obvious. For example, imagine the casino refuses payment to a player after the latter wins a game; unless special IT security measures are built in the on-line casino, the player might end up without any valid proof that might enable her to prove in court that she has been abused. In general, enough information on the development of the game should be logged so that posterior analysis of that information allows resolution of any dispute that may arise. Integrity and time-stamping of logfiles should be guaranteed, *i.e.* addition, deletion or modification of any log entry should be detectable; otherwise, the logfile owner could alter log entries at will. On the other hand, each log entry should be signed by the party having generated it, to prevent later repudiation.

1.1 Our Contribution

In this paper, we propose a method to shuffle and deal a deck of cards to the players that does not require any TTP, and allows e-gambling to reach standards of fairness, security and auditability similar to those common in physical gambling. Unlike for other TTP-free proposals, eliminating the TTP does not dramatically increase the computational complexity of our solution.

Casino games fall into three groups:

- Random draw games, with a single draw (*e.g.* dice, roulette) or with multiple draws (*e.g.* bingo, keno).

- Games where a value or a set of values are obtained in a non-secret way. Games where cards are visible (*e.g.* blackjack) fall into this category.
- Games where a value or a set of values are obtained in a secret way. Games where cards are reversed (*e.g.* poker) fall into this category.

There are several good solutions for the first and second categories. Two significant examples are [17] and [23]. Our contribution relates on the third category, which is the most complex one. Specifically, our method will be illustrated here on the poker game. Further detail and applications to other game categories can be found in patent [3].

Section 2 recalls previous work on secure e-gambling; special emphasis is made on mental poker, which is the name used in the literature for secure e-poker. Our solution is described in detail in Section 3. A security analysis considering various possible attacks is reported in Section 4. Section 5 is a conclusion.

2 Previous Work

Difficult problems have a special appeal. Mental poker is one such problem and has received substantial attention over the past two decades. In 1981, Shamir *et al.* [24] presented the first mental poker protocol, which is restricted to two players. In [21] and [5], it was shown that this protocol uses a cryptosystem allowing cards to be marked, which can leak information to the opponent and thus be fatal. In 1982, a new protocol was proposed in [14] which uses probabilistic encryption and prevents card marking, but is still limited to two players. Barany *et al.* proposed in 1983 a protocol for several players [1]. This protocol does not use cryptographic primitives and relies exclusively on permutations; its main drawback is that player confabulations are possible. Two years later, the problem of player confabulation was solved by Fortune and Merritt [12] using a TTP. In fact, most recent proposals [15,22,16,28,4] also use a TTP, because it is argued that using a TTP is the only way to obtain mental poker protocols which are both fair and reasonably efficient.

On the theoretical side, several solutions have been proposed, some of which do not require a TTP. However, those solutions share the drawback of requiring too much computation to be usable in practice. Along this line, Crépeau [6] presented in 1985 a mental poker protocol that minimizes the probability of successful player coalition, and one year later in [7] the same author presented an improved protocol that offers player confidentiality. This latter protocol uses a Zero-Knowledge-Proof (ZKP) subprotocol, which needs a substantial computing time. Later Kurosawa *et al.* in [18] and [19] and Schindelhauer in [25] also propose solutions that use ZKP. As an illustration of the amount of computing time needed by this kind of protocols, in [11] an implementation of the protocol [7] on three Sparc workstations is reported to have taken eight hours to shuffle a deck. Even though all the aforementioned proposals are very time consuming, [7,18,19] at least have the theoretical interest of not requiring disclosure of players' strategy after the game, a feature not offered by the protocol presented in this paper.

As mentioned above, we present a ZKP-free and TTP-free mental poker protocol which can be used in practical gaming scenarios, because it requires an amount of computation which is not much higher than the computation required by proposals using a TTP.

3 A Protocol Suite for E-gambling with Reversed Cards

In our protocol suite for e-games with reversed cards, deck shuffling is carried out in an efficient and fair way by the players themselves.

All players co-operate in shuffling, so that no player or player coalition can force a particular outcome, *i.e.* determine the card that will be obtained after shuffling. Every player generates a random permutation of the card deck and keeps it secret; the player then commits to her permutation using a bit commitment protocol. The shuffled deck is formed by the composition of all player permutations. The use of permutations to shuffle the deck was introduced in [1], and also was used in [6,7].

Note that reversing cards in the physical world translates to encrypting cards in e-gambling. Now, permuting (*i.e.* shuffling) encrypted cards requires encryption to be homomorphic, so that the outcome of permuting and decrypting (*i.e.* opening) a card is the same that would be obtained if the card had been permuted without prior encryption (*i.e.* reversal). The use of an additive homomorphic cryptosystem to shuffle the deck of cards and maintain the privacy of the cards was initially proposed in [18].

3.1 Notation

The following notation will be used in what follows:

- PL_i: i-th player
- $m_1|m_2$: Concatenation of messages m_1 and m_2.
- P_{entity}, S_{entity}: Asymmetric key pair of $entity$, where P_{entity} is the public key and S_{entity} is the private key.
- $S_{entity}\{m\}$: Digital signature of message m by $entity$, where digital signature means computing the hash value of message m using a collision-free one-way hash function and encrypting this hash value under S_{entity}.
- K_{entity}: Secret symmetric key of $entity$.
- $E_{K_{entity}}\{m\}$: Encryption of message m under K_{entity}.
- $D_{K_{entity}}\{c\}$: Decryption of message c under K_{entity}.

3.2 Card Representation and Permutation

In most e-gambling approaches, a prescribed ordering of cards in the deck is assumed, so that a card is represented by a scalar corresponding to its rank. In our protocol, a card representation is needed which allows card operations and permutations. Thus, we will map the usual scalar representation to a vector representation in the way described below.

Definition 1 (Card vector representation). *Let t be the number of cards in the deck. Let z be a prime number chosen by a player. A card can be represented as a vector*

$$v = (a_1, \cdots, a_t) \tag{1}$$

where there exists a unique $i \in \{1, \cdots, t\}$ for which $a_i \bmod z \neq 0$, whereas $\forall j \neq i$ it holds that $a_j \bmod z = 0$. The value of the card is i; assuming a prescribed ordering of cards, i is interpreted as a rank identifying a particular card.

The above vector representation for cards allows card permutations to be represented as matrices in a natural way.

Definition 2 (Card permutation matrix). *A permutation π over a deck of t cards is a bijective mapping that can be represented as a square matrix Π with t rows called* card permutation matrix*, where rows and columns are vectors of the form described by Expression (1):*

$$\Pi = \begin{pmatrix} \pi_{11} & \pi_{12} & \cdots & \pi_{1t} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \pi_{t1} & \pi_{t2} & \cdots & \pi_{tt} \end{pmatrix} \tag{2}$$

The i-th row of matrix Π is card $\pi(i)$, i.e. *the card resulting from applying permutation π to the card having rank i. Thus, all elements in the i-th row of Π are $0 \bmod z$ except π_{ij}, where $j = \pi(i)$.*

See Example 1 in Appendix A for an illustration of the above definition. The result below is straightforward from the above construction:

Proposition 1 (Permutation algebra). *With the above representation for cards and permutations, the result $w = \pi(v)$ of permuting a card v using a permutation π can be computed in vector representation as $w = v \cdot \Pi \pmod z$, where $\cdot$ denotes vector product. For this computation to work properly, the same value z must be used to represent v and Π.*

Example 2 in Appendix A illustrates the above proposition. Every player PL_i will use her own prime modulus z_i for representing and permuting cards. In order for PL_i to be able to operate her card permutation matrix with a card coming from another player PL_j, player PL_i must represent her permutation matrix using the modulus z_j corresponding to PL_j. This means transforming her permutation matrix based on z_i into an equivalent permutation matrix based on z_j, as defined below:

Definition 3 (Equivalent card permutation matrices). *Let Π and Π' be two $t \times t$ card permutation matrices using moduli z and z', respectively. Then Π and Π' are said to be equivalent if they represent the same permutation π of a*

set of t cards. Specifically, $\Pi = \{\pi_{ij}\}$ and $\Pi' = \{\pi'_{ij}\}$ are equivalent if and only if, $\forall i, j \in \{1, \cdots, t\}$, one has

$$\pi_{ij} \bmod z \neq 0 \Leftrightarrow \pi'_{ij} \bmod z' \neq 0$$

$$\pi_{ij} \bmod z = 0 \Leftrightarrow \pi'_{ij} \bmod z' = 0$$

3.3 Protocol Description

In order to meet the game auditability requirement, we introduce in this paper a new tool called distributed notarization chains (DNC). DNCs have a philosophy similar to the one of Lamport's hash chains [20] and their construction is described in Appendix B. Whenever a player builds a link of a DNC, she sends it the other players, so that all of them see the same DNC. Furthermore, each link is digitally signed by the player who built it, which guarantees authentication, integrity and non-repudiability for that link. A link can only be appended at the end of the DNC and no participant can modify or delete any link without being detected.

The initialization protocol is as follows:

Protocol 1 (Initialization)

1. *Each player PL_i is assumed to have an asymmetric key pair (P_i, S_i) whose public key has been certified by a recognized certification authority. Assume the card deck consists of t cards.*
2. *PL_i does:*
   (a) *Generate a permutation π_i of the card deck and keep it secret.*
   (b) *Generate a symmetric secret key K_i corresponding to a homomorphic cryptosystem allowing algebraic operations (additions and multiplications) to be carried out directly on encrypted data. Security requirements on this homomorphism are limited to resistance against ciphertext-only attacks; possible choices are [9,10].*
   (c) *Choose a prime value z_i which falls within the range of the cleartext space of the homomorphic cryptosystem used.*
   (d) *Build a link of the DNC which contains the value z_i used by PL_i. Link building can be done in parallel by all players, which expands the DNC at this point.*
   (e) *Build the card permutation matrix Π_i corresponding to π_i, using z_i.*
   (f) *Commit to this permutation Π_i using a bit commitment protocol [26]. Denote the resulting commitment by Cp_i.*
   (g) *Build the next link of the DNC following the previous expanded link. The new link contains the commitment Cp_i.*
   (h) *Choose s values $\{\delta_1, \cdots, \delta_s\}$ such that $\delta_j \bmod z_i = 0$, $\forall j \in \{1, \cdots, s\}$ and $s > t$.*
   (i) *Choose s values $\{\epsilon_1, \cdots, \epsilon_s\}$ such that $\epsilon_j \bmod z_i \neq 0$, $\forall j \in \{1, \cdots, s\}$ and $s > t$.*
   (j) *Homomorphically encrypt the previous values under the symmetric key K_i to get $d_j = E_{K_i}(\delta_j)$ and $e_j = E_{K_i}(\epsilon_j)$, $\forall j \in \{1, \cdots, s\}$.*

(k) *Build the next link of the DNC which contains the set* $\mathbf{D} = \{d_1, \cdots, d_s\}$.
(l) *Build another link of the DNC which contains the set* $\mathbf{E} = \{e_1, \cdots, e_s\}$.
(m) *Generate the vector representation for the t cards in the deck* $\{w_1, \cdots, w_t\}$ *and encrypt them under* K_i *using the aforementioned homomorphic cryptosystem to obtain* $w'_j = E_{K_i}(w_j)$.
(n) *Randomly permute the encrypted cards* $\{w'_1, \cdots, w'_t\}$.
(o) *Build the next link of the DNC which contains the card deck encrypted and permuted by* PL_i.

Once initialization is over, the player playing the croupier role performs chain contraction by building a link with a chaining value mixing together the last link created by every player PL_i. This link indicates that initialization is over and that the game can start.

Note that, even though each player encrypts the deck under her key during initialization, the deck of the game is a single one, namely a shuffled deck resulting from the composition of all players' secret permutations. Also, the initialization protocol results in each player PL_i publishing in the DNC her z_i and a commitment to her *secret* permutation Π. When PL_i wants a card, the following protocol is started:

Protocol 2 (Card draw)

1. PL_i *does:*
   (a) *Pick an integer value* v_0 *such that it falls within the range of cards in the deck,* i.e. $1 \leq v_0 \leq t$, *and which has not previously been requested. This operation is simple because it is public. All participants know the initial values chosen in previous steps.*
   (b) *Build the next link of the DNC which contains the vector representation* w_0 *of card value* v_0 *chosen by* PL_i.
2. PL_1 *does:*
   (a) *Check the validity of the link sent by* PL_i, *compute her equivalent card permutation* Π'_1 *for the modulus* z_i *published by* PL_i *in the DNC and permute* w_0 *to obtain* $w_1 = w_0 \cdot \Pi'_1$.
   (b) *Build the next link of the DNC, which contains* w_1 *and the name of the next player* PL_2 *in the computation.*
3. *For* $j = 2$ *to* $i - 1$, *player* PL_j *does:*
   (a) *Check the validity of the link sent by* PL_{j-1}, *compute her equivalent card permutation matrix* Π'_j *for the modulus* z_i *published by* PL_i *and permute* w_{j-1} *to obtain* $w_j = w_{j-1} \cdot \Pi'_j$.
   (b) *Build the next link of the DNC, which contains* w_j *and the name of the next player* PL_{j+1} *in the computation.*
4. *Player* PL_i *does:*
   (a) *Check the validity of the link sent by* PL_{i-1} *and permute* w_{i-1} *with her permutation matrix* Π_i *to obtain* $w_i = w_{i-1} \cdot \Pi_i$.
   (b) *Modify the m-th row of* Π_i *where* $m \in \{1, \cdots, t\}$ *is the value of card* w_{i-1}. *All values in the m-th row are changed to values that are nonzero modulo* z_i

(c) *Pick the encrypted card* w'_i *corresponding to clear card* w_i*. Note that the encrypted deck has been published in the last step of Protocol 1.*
(d) *Build the next link of the DNC, which contains* w'_i *and the name of the next player* PL_{i+1} *in the computation.*

5. *For* $j = i + 1$ *to* n*, player* PL_j *does:*
(a) *Check the validity of the link sent by* PL_{j-1} *and compute her equivalent card permutation matrix* Π'_j *for the modulus* z_i *published by* PL_i*. Use Protocol 3 below to encrypt* Π'_j *as* Π^c_j *under the key* K_i *corresponding to* PL_i*.*
(b) *Permute the encrypted card* w'_{j-1} *using her encrypted matrix* Π^c_j *to obtain* $w'_j = w'_{j-1} \cdot \Pi^c_j$*.*
(c) *Build the next link of the DNC, which contains* w'_j*. If* $j < n$*, the link also indicates the name of the next player* PL_{j+1} *in the computation.*

6. *When* PL_i *sees the link computed by* PL_n*, she does:*
(a) *Check the validity of the link computed by* PL_n*.*
(b) *Decrypt the card* w'_n *contained in the link computed by* PL_n *under her private key* K_i *to obtain the drawn card* $w_n = D_{K_i}(w'_n)$*. This finishes the card draw protocol.*

An extension of Protocol 2 for drawing several cards simultaneously (multi-card) is described in Appendix C. A loose end that remains is how does PL_j encrypt her permutation matrix Π'_j under PL_i's secret key K_i in Step (5a) of Protocol 2. To do that, player PL_j can only use the sets of encrypted values $\mathbf{D}$ and $\mathbf{E}$ published by PL_i in Step (2k) of Protocol 1. The specific procedure is described below:

Protocol 3 (Permutation matrix homomorphic encryption)

Let the homomorphic encryption of the cleartext matrix $\Pi_j = \{\pi_{kl}\}$ *under* K_i *be* $\Pi^c_j = \{\pi^c_{kl}\}$*. To compute* π^c_{kl}*, for* $1 \leq k, l \leq t$*, player* PL_j *does the following:*

1. *Generate a pseudorandom value* g*, such that* $1 \leq g \leq s$*, where* s *is the size of the sets* $\mathbf{D}, \mathbf{E}$ *of encrypted values published by* PL_i *in Protocol 1.*
2. *Randomly pick* g *values* $\{d_1, \cdots, d_g\}$ *of the set* $\mathbf{D}$ *and add them to obtain* $h = d_1 + \cdots + d_g$*. Remember that values in* $\mathbf{D}$ *are 0 modulo* z_i*, because they are homomorphically encrypted versions of values which are 0 modulo* z_i*, so by the homomorphic properties,* h *is also 0 modulo* z_i*.*
3. *Generate a pseudorandom value* c *such that* $c \bmod z_i \neq 0$ *and compute* $h' = c \cdot h$*.*
4. *If* $\pi_{kl} \bmod z_j = 0$*, then* $\pi^c_{kl} := h'$*.*
5. *If* $\pi_{kl} \bmod z_j \neq 0$*, then*
   (a) *Generate a pseudorandom value* g' *such that* $1 \leq g' \leq s$*.*
   (b) $\pi^c_{kl} := h' + e_{g'}$*, where* $e_{g'}$ *is the* g'*-th element of the set* E*.*

A player can also discard a card w by building a link of the DNC which says that the player is discarding a card and contains the encrypted version of the discarded card.

3.4 Game Validation

When a hand of the game is over, players should reveal their encryption keys and their permutations. The validation process is specified next:

Protocol 4 (Game validation)
Each player does the following:

1. *Check that the permutation revealed and used by each other player PL_i is the same permutation π_i to which she committed when publishing the commitment Cp_i in Protocol 1 (initialization). This check implies verifying the bit commitment for player PL_i.*
2. *Decrypt cards $\{w_1^{'}, \ldots, w_t^{'}\}$ published by each other player PL_i in the last step of Protocol 1 and check that the card deck is correct.*
3. *Use the private key K_i of each other player PL_i to decrypt the result of permuting encrypted cards at Step (5b) of Protocol 2. Check that permutations were correctly performed.*
4. *Check that cards discarded by other players have not been used during the game.*
5. *If necessary, use the DNC to prove any detected misbehaviors by any other player to a third-party (casino, court, etc.).*

4 Security Analysis

Let us examine a collection of possible attacks and check that they fail:

A coalition wants to get the cards drawn by a player Player PL_i draws one or more cards in an instance of Protocol 2. In subsequent instances belonging to the same hand, a coalition of players attempt to determine the cards drawn by PL_i. To do that, the coalition must construct a (multi)card (see Appendix C) with the card value(s) drawn by PL_i and encrypt that (multi)card. However, the coalition will not get PL_i's card(s) because, if a card is requested which had been previously requested, what is obtained is a vector with all components different from 0 modulo z_i. This is due to the modification of matrix Π_i at Step (4b) of Protocol 2. The resulting multicard has all components different from 0 modulo z_i and is thus a non-valid multicard (by definition, see Appendix C).

A player uses a wrong modulus Assume that, to permute a card coming from player PL_{k-1}, player PL_k computes her equivalent permutation matrix Π_k' using $z_j \neq z_{k-1}$. This is detected by PL_i in the last step of Protocol 2, because w_n' decrypts into a non-valid card (with too many nonzero components modulo z_i). Player PL_i has no option other than reporting the wrong decryption; otherwise PL_i would not be able to show her cards during game validation. Verification performed during Protocol 4 discloses the identity of the cheater PL_k.

A player does not use the permutation she committed to A player may choose to use a permutation Π different from the one she committed to during initialization.

If the player changes her permutation for the whole game, this is detected during game validation. If the player changes her permutation only during some parts of the game, two things may happen:

- Some of the other players get duplicated cards. A player getting a duplicated card is forced to report it (otherwise she will not be able to show her cards during game validation). Upon such report, the game is stopped and game validation started.
- The change is not detected during the game. In this case, it is detected during game validation and the dishonest player is identified.

A player supplies wrong sets D or E If, during initialization, a player does not supply correct sets $\mathbf{D} = \{d_1, \cdots, d_s\}$ or $\mathbf{E} = \{e_1, \cdots, e_s\}$, then permutations of encrypted cards cannot be correctly computed. Two things can happen:

- The card obtained by PL_i at the end of Protocol 2 is not valid. In this case, PL_i is forced to report the problem.
- The card obtained by PL_i is valid. In this case, the problem is detected during game validation.

A player supplies a wrong encrypted deck The game validation protocol verifies that all players have supplied correct encrypted decks during initialization.

A player builds an incorrect DNC link During execution of Protocol 2, each player checks that the previous link of the DNC has correctly been built. Any wrong link is reported (all players see all links so it is risky not to report a wrong link when discovered).

A player requests a card which had already been requested Cards requested at the beginning of Protocol 2 are recorded in the DNC. If a player requests a card that had already been requested, this is detected by the rest of players (all of them see the DNC links).

A player does not correctly encrypt her permutation An incorrectly encrypted card permutation matrix can yield non-valid or duplicated cards, which is detected during the game. Even if all cards are valid, a wrongly encrypted permutation is detected during game validation.

Player withdrawal Depending on when a player withdraws from the game, different things happen:

- If a player withdraws during initialization, the game simply proceeds without her.
- If a player withdraws immediately after initialization, the game proceeds without her. In this case, the composition of permutations must be computed without using the permutation of the withdrawn player.
- If a player withdraws during the game (*i.e.* during Protocol 2), the game stops at the moment of withdrawal. Players show their cards and game validation is started. If the player has withdrawn without justification, she may be fined.

5 Conclusion

A solution for obtaining impartial random values in on-line gambling has been presented in this paper. Unlike most previous proposals, our method does not require any TTP and allows e-gambling to reach levels of fairness, security and auditability similar to those common in physical gambling. In addition, eliminating the TTP does not result in a dramatical increase of computation.

The solution has been specified for the particular case of games with reversed cards (*e.g.* poker), but it can be easily adapted for games with open cards (*e.g.* blackjack) and for random draw games (*e.g.* keno).

Appendix A. Examples of Card Representation and Permutation

Example 1. Let $t = 5$ and consider the following permutation of five cards

$$\pi = \begin{pmatrix} 3\ 5\ 4\ 1\ 2 \end{pmatrix} \tag{3}$$

If $z = 5$ is taken, a possible matrix representation of π is as follows

$$\Pi = \begin{pmatrix} 5 & 10 & 3 & 5 & 15 \\ 25 & 10 & 35 & 5 & 6 \\ 50 & 10 & 10 & 8 & 5 \\ 4 & 60 & 5 & 50 & 25 \\ 15 & 7 & 35 & 60 & 10 \end{pmatrix} \tag{4}$$

Note that, in the first row, the only element which is nonzero modulo 5 is the third one, that is, the one in the position corresponding to $\pi(1) = 3$. Similarly, in the second row, the only nonzero element modulo 5 is in the 5-th position, because $\pi(2) = 5$. And so on for the other rows.

Example 2. Using $t = 5$, $z = 5$ and the permutation matrix of Example 1, the card $v = (10, 15, 8, 20, 20)$ (vector representation modulo 5 of a card with value 3) is permuted as

$$w = (10, 15, 8, 20, 20) \cdot \Pi = (1205, 1670, 1435, 2389, 980) \tag{5}$$

Since the only component of the permuted card w that is nonzero modulo z is the fourth one, w is the vector representation of a card with value 4 (the fourth card of the ranked deck). In this way, $\pi(3) = 4$, which is consistent with the fact that the only component that is nonzero modulo z in the third row of the permutation matrix is the fourth one.

Appendix B. Distributed Notarization Chains (DNCs)

Each operation performed during the e-game will be notarized as a link of a distributed notarization chain (DNC). DNCs are efficiently computable and consist of links which are constructed and chained as follows:

Link structure Every link m_k of the DNC is formed by two fields: a data field D_k and a chaining value X_k. The data field D_k consists of three subfields:
- Timestamp T_k, which contains the link generation time according to the clock of the participant who generated the link (synchronizing the clocks of all participants is not needed).
- Link subject or concept C_k, which describes the information contained in the link, *e.g.* a step in the game, a commitment or an outcome.
- Additional attributes V_k, which depend on the subject C_k. For a link corresponding to a commitment, V_k will contain the encrypted commitment.

Link chaining Chaining is guaranteed by chaining values X_k included in each link. First, the chaining value X_{k-1} of the previous link is concatenated with the data field D_k of the current link; then the hash value of the concatenation is computed and signed with the private key of the author of the k-th link, *i.e.*

$$X_k = S_{author}\{D_k | X_{k-1}\} \tag{6}$$

There are moments during the game at which operations do not need to be sequential, but can be carried out in parallel by the participants. Parallel execution can be accomodated in the distributed notarization system by introducing two operations:

Chain expansion Parallel execution can be notarized by *expanding the DNC*, whereby participants $\{PL_1, \cdots, PL_n\}$ independently compute their chaining values $X_k^{PL_1}, \cdots, X_k^{PL_n}$ using the chaining value X_{k-1} of the previous link:

$$\begin{aligned} X_k^{PL_1} &= S_{PL_1}\{D_k^{PL_1} | X_{k-1}\} \\ &\vdots \\ X_k^{PL_n} &= S_{PL_n}\{D_k^{PL_n} | X_{k-1}\} \end{aligned} \tag{7}$$

Chain contraction It happens when the protocol requires sequential execution after a stage of expansion where n participants have computed links in parallel. A single link is obtained which is chained to previous links. The participant initiating the sequential execution concatenates its data field D_k with all chaining values of previous links $\{X_{k-1}^{PL_1} | \ldots | X_{k-1}^{PL_n}\}$, computes the hash of the concatenation and signs it to obtain the chaining value X_k of the first sequential link:

$$X_k = S_{PL_i}\{D_k | X_{k-1}^{PL_1} | \cdots | X_{k-1}^{PL_n}\} \tag{8}$$

The following properties of a DNC make it a good tool for distributed notarization:

- The DNC is not possessed by a single participant, but by all of them. Whenever a participant builds a link of a DNC, she sends it to the other participants, so that all of them see the same DNC. If a participant recomputes the chain to add false links to it, the manipulated chain will not match the copies held by the rest of participants, and manipulation will be detected.
- If someone deletes or modifies one or more links, the chain will show an inconsistency at the point of deletion.
- The chaining and the structure of links allow the exact sequence and time of link construction to be securely determined.
- As links are signed by the participant who built them, link authorship can be securely determined.
- Link computation is performed in parallel whenever the protocol allows it (using the expansion operation). This results in improved performance without degrading security.

Appendix C. Extensions

In each run of Protocol 2, player PL_i gets only one card. The protocol can be extended so that the player draws several cards in a single protocol run. To do this, let us define a way to pack several cards together:

Definition 4 (Multicard). *Given a deck with t cards and a prime value z, a multicard ξ is a vector of t elements*

$$\xi = (a_1, \cdots, a_t) \tag{9}$$

where there are up to $M < t$ components a_i, such that $a_i \bmod z \neq 0$. The index i of each component a_i that is nonzero modulo z represents one of the cards contained in the multicard. By convention, a multicard with more than M components that are nonzero modulo z is not valid.

Protocol 2 can be adapted to multicards as explained below:

- At Step (1a) of Protocol 2, PL_i should choose the cards she wishes. Let the vector representation of these be $w_0^1, \cdots, w_0^x$, with $x < M$.
- A multicard $\xi_0 = \sum_{j=1}^{x} w_0^j$ is obtained by adding the chosen cards.
- At Step (1b), the DNC link would be computed using the multicard ξ_0 rather than a single card w_0.
- The protocol carries on until Step (4b), where all rows corresponding to values of cards in the multicard are modified.
- The protocol then proceeds as described above. In the final Step (6b), PL_i decrypts the card computed by PL_n and obtains a multicard.

Acknowledgments

The second author is partly supported by the Spanish Ministry of Science and Technology and the European FEDER fund under project TIC2001-0633-C03-01 "STREAMOBILE". Thanks go to Jordi Herrera for useful comments on some parts of this work.

References

1. I. Barany and Z. Furedi, "Mental poker with three or more players", Technical report, Mathematical Institute of the Hungarian Academy of Sciences, 1983.
2. M. Blum, "Coin flipping by telephone: A protocol for solving impossible problems", in *Proceedings of the 24th IEEE Computer Conference (CompCon.)*, pp. 175-193, 1982.
3. Jordi Castellà-Roca, Andreu Riera-Jorba, Joan Borrell-Viader and Josep Domingo-Ferrer, "A method for obtaining an impartial result in a game over a communications network and related protocols and programs", international patent PCT ES02/00485, Oct. 14, 2002.
4. J. S. Chou and Y. S. Yeh, "Mental poker game based on a bit commitment scheme through network", *Computer Networks*, vol. 38, pp. 247-255, 2002.
5. D. Coppersmith, "Cheating at mental poker", in *Advances in Cryptology - Crypto '85* (ed. H. C. Williams), LNCS 218, Berlin: Springer Verlag, pp. 104-107, 1986.
6. C. Crépeau, "A secure poker protocol that minimizes the effect of player coalitions", in *Advances in Cryptology - Crypto '85* (ed. H. C. Williams), LNCS 218, Berlin: Springer Verlag, pp. 73-86, 1986.
7. C. Crépeau, "A zero-knowledge poker protocol that achieves confidentiality of the players' strategy or how to achieve an electronic poker face", in *Advances in Cryptology - Crypto'86* (ed. A. M. Odlyzko), LNCS 263, Berlin: Springer-Verlag, pp. 239-250, 1986.
8. R. DeMillo and M. Merritt, "Protocols for data security", *Computer*, vol. 16, pp. 39-50, 1983.
9. J. Domingo-Ferrer, "A new privacy homomorphism and applications", *Information Processing Letters*, vol. 60, pp. 277-282, 1996.
10. J. Domingo-Ferrer, "A provably secure additive and multiplicative privacy homomorphism", in *Information Security* (eds. A. Chan and V. Gligor), LNCS 2433, pp. 471-483, 2002.
11. J. Edwards, *Implementing Electronic Poker: A Practical Exercise in Zero-Knowledge Interactive Proofs*. Master's thesis, Department of Computer Science, University of Kentucky, 1994.
12. S. Fortune and M. Merritt, "Poker protocols", in *Advances in Cryptology: Proceedings of Crypto'84* (eds. G. R. Blakley and D. Chaum), LNCS 196, Berlin: Springer-Verlag, pp. 454-466, 1985.
13. Gambling Review Body, Department for Culture Media and Sport of Great Britain, chapter 13, page 167, July 17, 2001. `http://www.culture.gov.uk/role/gambling_review.html`,
14. S. Goldwasser and S. Micali, "Probabilistic encryption & how to play mental poker keeping secret all partial information", in *Proceedings of the 18th ACM Symposium on the Theory of Computing*, pp. 270-299, 1982.

15. C. Hall and B. Schneier, "Remote electronic gambling", in *13th ACM Annual Computer Security Applications Conference*, pp. 227-230, 1997.
16. L. Harn, H. Y. Lin and G. Gong, "Bounded-to-unbounded poker game", *Electronics Letters*, vol. 36, pp. 214-215, 2000.
17. M. Jakobsson, D. Pointcheval and A. Young, "Secure mobile gambling", in *Topics in Cryptology - CT-RSA 2001*, (ed. D. Naccache), LNCS 2020, Springer-Verlag, pp. 100-109, San Francisco, CA, USA, April 8-12, 2001.
18. K. Kurosawa, Y. Katayama, W. Ogata and S. Tsujii, "General public key residue cryptosystems and mental poker protocols", in *Advances in Cryptology - EuroCrypt'90* (ed. I. B. Damgaard) LNCS 473, Berlin: Springer-Verlag, pp. 374-388, 1990.
19. K. Kurosawa, Y. Katayama and W. Ogata, "Reshufflable and laziness tolerant mental card game protocol", *TIEICE: IEICE Transactions on Communications/Electronics/Information and Systems*, vol. E00-A, 1997.
20. L. Lamport, "Password authentication with insecure communications", *Communications of the ACM*, vol. 24, pp. 770-771, 1981.
21. R. Lipton, "How to cheat at mental poker", in *Proc. AMS Short Course on Cryptography*, 1981.
22. R. Oppliger and J. Nottaris, "Online casinos", in *Kommunikation in verteilten Systemen*, pp. 2-16, 1997.
23. E. Kushilevitz and T. Rabin, "Fair e-lotteries and e-casinos", in *Topics in Cryptology CT-RSA 2001*, (ed. D. Naccache), LNCS 2020, Springer-Verlag pp. 110-119, San Francisco, CA, USA, April 8-12, 2001.
24. A. Shamir, R. Rivest and L. Adleman, "Mental poker", *Mathematical Gardner*, pp. 37-43, 1981.
25. C. Schindelhauer, "A toolbox for mental card games", Medizinische Universität Lübeck, 1998. `http://citeseer.nj.nec.com/schindelhauer98toolbox.html`
26. B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C, 2nd Edition*, New York: Wiley, 1996.
27. M. Yung, "Cryptoprotocols: Subscription to a public key, the secret blocking and the multi-player mental poker game", in *Advances in Cryptology: Proceedings of Crypto'84* (eds. G. R. Blakley and D. Chaum), LNCS 196, Berlin: Springer-Verlag pp. 439-453, 1985.
28. W. Zhao, V. Varadharajan and Y. Mu, "Fair on-line gambling", in *16th IEEE Annual Computer Security Applications Conference (ACSAC'00)*, New Orleans, Louisiana, pp. 394-400, 2000.