Privacy Homomorphisms for E-Gambling and Mental Poker

Jordi Castellà-Roca, Vanesa Daza, Josep Domingo-Ferrer, *Senior Member, IEEE,* and Francesc Sebé

***Abstract*— With the development of computer networks, situations where a set of players remotely play a game (*e-gaming*) have become usual. Often players play for money (*e-gambling*), which requires standards of security similar to those in physical gambling. Cryptographic tools have been commonly used so far to provide security to e-gambling. Homomorphic encryption is an example of such tools.**

In this paper we review the *mental poker* protocols, where players are assumed to remotely play poker. We focus on the key advantage of using cryptosystems with homomorphic properties (*privacy homomorphisms*) because they offer the possibility of manipulating cards in encrypted form.

***Index Terms*— Cryptography, privacy homomorphism, mental poker**

I. Introduction

The growth of computer networks has allowed many activities that used to require physical presence to become doable over the network. One example is e-gambling. In this paper we focus on mental poker, *i.e.* a poker game played among players that are not physically together but communicate through computer networks.

A mental poker system needs to provide protocols to generate a deck of cards, shuffle it and confidentially deal the cards to players. Naive approaches assume the existence of a trusted third party (TTP), who performs some or all of the aforementioned operations [1].

Since the assumption on all players trusting this central node is not always realistic, research on solutions not requiring a trusted party has become a hot topic [2]–[5]. To ensure the players a fair play, standards of security similar to those in physical gambling must be guaranteed. For instance, it is good for trust generation that all players participate in the computation of the shuffled deck of cards. Current proposals in the literature are based on a paradigm consisting of the following steps:

1) The players generate the deck of 52 face-up cards $D = \{d_1, \ldots, d_{52}\}$. All players agree on which card each d_i corresponds to.
2) The elements of set D are encrypted using some cryptosystem. The result is the encrypted (or face-down) deck of cards $C = \{c_1, \ldots, c_{52}\}$, $c_i = E(d_i)$.
3) The encrypted deck of cards is processed sequentially by all the players. During this process, the player receives the set C, permutes its elements and then re-masks (*i.e.* re-encrypts) them. The set resulting from the previous step corresponds to the shuffled encrypted deck of cards.

The shuffled encrypted deck of cards is made up of 52 encrypted values such that none of the players knows which card is contained in each. Note that if Step 3 consisted of a single permutation, at least the player who performed Step 2 would know the content of the final shuffled and encrypted deck since she knows which cryptogram corresponds to each card. For a secure generation of the encrypted shuffled deck, it is necessary that, after permuting the encrypted deck, its elements are re-masked. Re-masking consists of performing some operation over each encrypted value so that its cleartext does not change but the re-masked message is not linkable to the same message before re-masking.

In this paper we survey the homomorphic properties of the cryptosystems used in current state-of-the-art mental poker protocols.

The rest of the paper is distributed as follows. Section II reviews the concept of privacy homomorphism and gives some examples. Section III surveys the state of the art of TTP-free mental poker protocols and how homomorphims play a key role in them. Finally, Section IV is a conclusion.

II. Privacy homomorphism

Privacy homomorphisms (PH) were introduced by Rivest, Adleman and Dertouzos in [6]. A PH is an encryption function which maps a set F of operations on plaintext to another set F' of operations on ciphertext. Operations on encrypted data are computed without knowledge of the decryption key.

Privacy homomorphisms are used as building blocks in secure computation applications such as computing delegation, data delegation and e-gambling: the idea is to encrypt data at a classified level, to process them at an unclassified level and to decrypt the result at a classified level.

Next we describe some examples of privacy homomorphisms.

A. Homomorphic properties of ElGamal cryptosystem

The ElGamal cryptosystem is a famous public key cryptosystem described in [7]. It is defined over a group $G \subset \mathbb{Z}_p^*$ of order $q|p-1$, where p and q are large primes. Let $g \in \mathbb{Z}_p^*$ be a generator of G.

The secret key corresponds to some secret value $x \in \mathbb{Z}_q$ and the public key is $h = g^x \bmod p$.

The authors are partly supported by the Catalan Government under grant 2005 SGR 00446, and by the Spanish Ministry of Science and Education through project SEG2004-04352-C04-01 "PROPRIETAS".

The authors are with the Rovira i Virgili University of Tarragona, Dept. of Computer Engineering and Maths, Av. Països Catalans, 26, E-43007 Tarragona, Catalonia (e-mail: {jordi.castella, vanesa.daza, josep.domingo, francesc.sebe}@urv.net)

A message $m \in \mathbb{Z}_p^*$ is encrypted by generating a random value $t \in \mathbb{Z}_q$ and then computing $E(m) = (c, c') = (mh^t, g^t)$.

The security of ElGamal relies on the so-called Decisional Diffie-Hellman (DDH) assumption, which states that the following two ensembles $(g, g^{t_1}, g^{t_2}, g^{t_3})$ and $(g, g^{t_1}, g^{t_2}, g^{t_1 t_2})$ are computationally indistinguishable, where t_1, t_2 and t_3 are chosen at random in $\mathbb{Z}_q$.

The ElGamal cryptosystem is a multiplicative privacy homomorphism. Let us assume two messages m_1 and m_2 encrypted under the same public key h, *i.e.* $E(m_1) = (c_1, c_1') = (m_1 h^{t_1}, g^{t_1})$ and $E(m_2) = (c_2, c_2') = (m_2 h^{t_2}, g^{t_2})$. By computing $E(m_1)E(m_2) = (c_1 c_2, c_1' c_2')$, we obtain $(m_1 m_2 h^{t_1} h^{t_2}, g^{t_1} g^{t_2}) = (m_1 m_2 h^{t_1+t_2}, g^{t_1+t_2})$, which corresponds to $E(m_1 m_2)$.

B. Homomorphic properties of Kurosawa et al. general public key cryptosystem

Kurosawa *et al.* presented in [2] a public key cryptosystem based on the r^{th} residue problem. In this cryptosystem, the secret key is a pair of large prime numbers (p, q) and the public key parameters are (n, r, y), where $n = p\,q$ and r and y are elements satisfying some technical requirements detailed in [2].

Then, in order to encrypt a plaintext m, $0 \leq m < r$, a number t is chosen at random and the value $E(m) = y^m t^r \bmod n$ is computed thereafter.

This cryptosystem is an additive privacy homomorphism. That is, let m_1 and m_2 be two messages encrypted using the same public key (n, r, y), *i.e.* $E(m_1) = y^{m_1} t_1^r \bmod n$ and $E(m_2) = y^{m_2} t_2^r \bmod n$, (for some randomly chosen t_1 and t_2); then it holds that $E(m_1 + m_2 \bmod r) = E(m_1)E(m_2) \bmod n$.

Note that $E(m_1)E(m_2) = y^{m_1} {t_1}^r y^{m_2} {t_2}^r = y^{m_1+m_2}(t_1 t_2)^r = y^{m_1+m_2} t_3^r = E(m_1 + m_2 \bmod r)$.

C. Domingo-Ferrer's 1996 algebraic privacy homomorphism

The PH proposed in [8] is additive and multiplicative. The public parameters are δ and n, where δ is a security parameter and n is the product of two sufficiently large primes p and q. The secret key consists of the following data p, q, $t_p \in \mathbb{Z}_p$ and $t_q \in \mathbb{Z}_q$, where t_p and t_q generate a large multiplicative subgroup in $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$ respectively.

A message $m \in \mathbb{Z}_n$ is encrypted with the following steps. First, m is randomly split into shares $s_1, \ldots, s_\delta$ such that $m = \sum_{j=1}^{\delta} s_j \bmod n$ and $s_j \in \mathbb{Z}_n$. Next, m is encrypted by computing $E(m) = ([s_1 \cdot t_p \bmod p, s_1 \cdot t_q \bmod q], [s_2 \cdot t_p^2 \bmod p, s_2 \cdot t_q^2 \bmod q], \ldots, [s_\delta \cdot t_p^\delta \bmod p, s_\delta \cdot t_q^\delta \bmod q])$.

The ciphertext can be seen as a polynomial, so the addition and subtraction of ciphertext can be done component-wise over $\mathbb{Z}$, *i.e.* adding terms of the polynomials with the same degree. The multiplication works in the same way. All terms are cross-multiplied in $\mathbb{Z}_n$, with a j_1-th degree term by a j_2-th degree term yielding a $(j_1 + j_2)$-th degree term; finally, terms having the same degree are added up. This privacy homomorphism is secure against ciphertext-only attacks.

D. Domingo-Ferrer's 2002 algebraic privacy homomorphism

Another PH preserving the addition and multiplication is presented in [9]. Public parameters are a positive integer $\delta > 2$ and a large integer N. N should have many small divisors and at the same time there should be many integers less than N that can be inverted modulo N. The secret parameters are $y \in \mathbb{Z}_N$ such that there exists an inverse $y^{-1} \bmod N$ and a small divisor $N' > 1$ of N. The secret key is (y, N').

The encryption of a message $m \in \mathbb{Z}_{N'}$ is as follows. First, m is randomly split into secret shares $s_1, \ldots, s_\delta$ such that $m = \sum_{j=1}^{\delta} s_j \bmod N'$ and $s_j \in \mathbb{Z}_{N'}$. The encrypted message is:

$$E(m) = (s_1 \cdot y \bmod N, \ldots, s_\delta \cdot y^\delta \bmod N)$$

Addition and multiplication are performed in a way similar as described in Section II-C. This privacy homomorphism is secure against ciphertext-only attacks.

III. Mental poker protocols based on homomorphic encryption

Mental poker is played like ordinary poker but without physical elements (like cards) nor verbal communication; all exchanges between players must be accomplished using messages [10]. Any player may try to cheat.

A mental poker protocol must guarantee the fairness of the game and, if a player tries to cheat, the protocol must detect or avoid the cheating. In [11], Crépeau enumerated the requirements and properties that must be met by a mental poker protocol.

Some mental poker protocols require the disclouse of the secret information at the end of the game to verify the game fairness. Nonetheless, the secret information also reveals the strategy of players. The most advanced protocols use zero-knowledge proofs to ensure the honesty of the game without revealing the strategies of players.

A zero-knowledge proof is an interactive and probabilistic method for a player (*prover*) to efficiently prove to another player (*verifier*) that a statement is true, without conveying any additional knowledge other than the statement. The wide applicability of zero-knowledge proofs is due to the fact that they can be used to guarantee honest behavior while maintaining privacy. They were introduced in [12].

Zero-knowledge proofs must satisfy the following three properties:

1) Completeness: if the statement is true, an honest verifier (that is, one following the protocol properly) will be convinced of this fact by an honest prover.
2) Soundness: if the statement is false, no cheating prover can convince an honest verifier that it is true, except with some small probability.
3) Zero-knowledge: if the statement is true, no cheating verifier learns anything other than this fact. This is formalized by showing that every cheating verifier has some simulator that, given only the statement to be proven (and no access to the prover), can produce a transcript that "looks like" an interaction between an honest prover and the cheating verifier.

Next we describe some current state-of-the-art mental poker protocols by focusing on how they use homomorphic cryptosystems.

A. *Kurosawa* et al. *mental poker protocols*

In [2], Kurosawa *et al.* presented a mental poker protocol that uses their above-described homomorphic public-key cryptosystem based on the r^{th} residue problem. This protocol satisfies all the aforementioned security requirements.

Before the game begins, every player computes a key pair for the cryptosystem, and her public key is sent to all players in the protocol. A face-up card of value d_j is defined as an l-tuple, where l is the number of players. Components of the tuple are shares of the value d_j corresponding to a linear secret sharing scheme (the authors of [2] considered Karnin, Greene and Hellman's secret sharing scheme [13] and Shamir's secret sharing scheme [14] but the protocol is actually generalizable to any linear secret sharing scheme). Then, every player is appropriately related to a component of the tuple. Without loss of generality we assume that the i-th component belongs to player $\mathcal{P}_i$.

Turning a card d_j face down is performed by encrypting every component of the tuple using Kurosawa *et al.*'s homomorphic public-key cryptosystem. More specifically, every component is encrypted by using the public key of the player associated with this component. In this way, every player can decrypt only the part of the card associated to him.

To re-mask a card of the deck, every player computes an l-tuple of shares of the value 0 using the designated linear secret sharing scheme. Again, every component is encrypted using the public key of the player related to that component. By multiplying component-wise both encrypted tuples, we get the re-masked card.

Finally, shuffling the deck is performed sequentially by all players by computing a permutation of all tuples in the deck and re-masking each of the resulting tuples.

B. *Efficient mental card shuffling via optimized arbitrary-sized Benes permutation network*

The proposal presented in [3] uses optimized arbitrary-sized Benes permutation networks (OAS-Benes PN) to shuffle the deck of cards. Benes PN are directed graphs of bounded degree with the same number of inputs and outputs. A Benes PN is capable of realizing all possible input permutations with a set of disjoint edge paths. It is constructed from 2×2 switches. We note the two inputs of the switch as $\{I_1, I_2\}$ and the two outputs as $\{O_1, O_2\}$. The binary control of the switch determines whether $(O_1 = I_1, O_2 = I_2)$ or $(O_1 = I_2, O_2 = I_1)$.

The inputs of the switches are encrypted using the ElGamal public key cryptosystem. When the encrypted messages go through the switch, they are re-masked using the homomorphic properties of ElGamal. Since ElGamal is a multiplicative homomorphism, a cryptogram can be re-masked by multipliying it by an encryption of 1. The zero-knowledge proofs presented in [15] are used to prove correct re-masking.

The use of the OAS-Benes PN is distributed among the l players. $T + 1$ OAS-Benes PN are used in order to assure an operational robustness against collusion by T cheating players. T is computed according to the number of players l, using the following expression $T < \left\lfloor \frac{l-1}{2} \right\rfloor$. So, the maximum number of cheaters is 0 when $l \in \{2, 3, 4\}$ and 1 when $l = 5$.

In the case of mental poker an OAS-Benes PN consists of 249 switches. The protocol establishes that there are $T + 1$ OAS-Benes PN for sufficient fault tolerance, so the total number of switches is $249(T + 1)$. The switches are grouped in stages. The OAS-Benes PN used in the mental poker protocol have 9 stages, and each stage has an average of $249/9$ switches. Each player is thereby assigned an average of $\left(\frac{T+1}{l}\right)(2\lfloor \log 52 \rfloor - 1)$ adjacent stages. In the case of mental poker, a player computes $\frac{(T+1)9}{l}$ stages or $249(\frac{T+1}{l})$ switches.

C. *Practical mental poker without a TTP based on homomorphic encryption*

The mental poker protocol presented in [4] uses a new representation for cards and permutations. This representation allows players to draw a new card by computing a permutation of a value in an encrypted way. In this way, the new card is only known to the player who requested it.

In most mental poker protocols, a prescribed ordering of cards in the deck is assumed, so that a card is represented by a scalar corresponding to its rank. In [4], a card representation is needed which allows card operations and permutations. Thus, the usual scalar representation is mapped to a vector representation in the way described below.

Let p be a prime number chosen by a player. A card can be represented as a vector

$$d_i = (d_{i,1}, \cdots, d_{i,52}) \tag{1}$$

where there exists a unique $j \in \{1, \cdots, 52\}$ for which $d_{i,j} \bmod p \neq 0$, whereas $\forall k \neq j$ it holds that $d_{i,k} \bmod p = 0$. The value of the card is j; assuming a prescribed ordering of cards, j 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.

A permutation π over a deck of 52 cards is a bijective mapping that can be represented as a square matrix Π with 52 rows called *card permutation matrix*, where rows and columns are vectors of the form described by Expression (1):

$$\Pi = \begin{pmatrix} \pi_{1,1} & \pi_{1,2} & \cdots & \pi_{1,52} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \pi_{52,1} & \pi_{52,2} & \cdots & \pi_{52,52} \end{pmatrix} \tag{2}$$

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

With the above representation for cards and permutations, the result $w = \pi(d_i)$ of permuting a card d_i using a permutation π can be computed as $w = d_i \cdot \Pi$, where $\cdot$ denotes vector

product. For this computation to work properly, the same value p must be used for d_i and Π.

Using the previous notation, the operation of the protocol can be described as follows: in the set-up phase, each player generates 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. The preferred choice is [9], because it is secure against ciphertext-only attacks. An alternative choice is [8].

The deck of cards is shuffled as follows. 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.

$\mathcal{P}_i$ draws a card using the following steps. She picks a value d_i in D that nobody else has picked before, and she gets her card by computing $\pi_n \circ \cdots \circ \pi_2 \circ \pi_1(d_i)$. But, since permutations are kept secret, the player must use a special trick in order to get her value. Thus, $\mathcal{P}_i$ gets the values $w' = \pi_{i-1}(\cdots(\pi_2 \circ (\pi_1(d_i))))$ in the clear. Next, $\mathcal{P}_i$ secretly computes $w'' = \pi_i(w')$ and encrypts w'' using the above homomorphic cryptosystem. $\mathcal{P}_i$ sends the w'' encryption to $\mathcal{P}_{i+1}$. $\mathcal{P}_{i+1}$ encrypts her permutation π_{i+1} using the key K_i and applies her encrypted permutation to the encrypted value sent by $\mathcal{P}_i$. Then $\mathcal{P}_{i+1}$ obtains $\pi_{i+1}(w'')$ in encrypted form (so that only $\mathcal{P}_i$ can obtain $\pi_{i+1}(w'')$ in the clear) and $\mathcal{P}_{i+1}$ sends the resulting cryptogram to $\mathcal{P}_{i+2}$. The rest of players, $\mathcal{P}_{i+2}, \ldots, \mathcal{P}_n$, do the same steps as $\mathcal{P}_{i+1}$. Finally, $\mathcal{P}_n$ sends $\pi_n \circ \cdots \circ \pi_2 \circ \pi_1(d_i)$ encrypted with K_i to $\mathcal{P}_i$. $\mathcal{P}_i$ decrypts it and obtains her card.

D. The Barnett-Smart mental poker protocol

The Barnett-Smart [5] protocol suite can be implemented using the ElGamal [7] encryption, or the Paillier probabilistic encryption function [16]. We describe the ElGamal version of the protocol suite.

If there are l players, $\{\mathcal{P}_1, \ldots, \mathcal{P}_l\}$, they set up the system as follows. All players agree on a prime number p, where $p = 2q + 1$ and q is an odd prime number. The computations are done over $\mathbb{Z}_p$. They also agree on a generator $g \in \mathbb{Z}_q$. Next, each player $\mathcal{P}_i$ generates a random private key x_i, and she publishes $h_i = g^{x_i}$. Finally, players compute the public key $h = \prod_{i=1}^{l}(h_i)$.

Players shuffle the deck of cards with the following steps. In the first step all players agree on the values to be used to represent the deck of cards. These values are the deck of face-up (cleartext) cards. The value of each card is public. Players compute the face-down (encrypted) deck of cards. Cards are encrypted under the public key h using 1 instead of a random factor in ElGamal encryption. Next, all players permute and re-mask the face-down deck of cards as described in [3], using ElGamal re-masking. Finally each player proves in zero-knowledge that she has re-masked and permuted the deck of cards properly. The zero-knowledge proof used is based on the one presented in [17].

IV. Conclusions

Trusted third parties greatly facilitate and simplify many processes that are carried out over the Internet. However, the availability of TTPs is not always realistic (there are not many entities trusted by everybody). For the case of mental poker, we have reviewed protocols which eliminate the need for a TTP by using privacy homomorphisms. Indeed, we have shown how homomorphic encryption can be used to manipulate cards in encrypted form, which allows players to manage cards co-operatively. Such a co-operation generates more trust among players than a conventional TTP.

References

[1] C. Hall and B. Schneier, "Remote electronic gambling," in *13th Annual Computer Security Applications Conference*. ACM, December 1997, pp. 227–230.

[2] K. Kurosawa, Y. Katayama, W. Ogata, and S. Tsujii, "General public key cryptosystems and mental poker protocols," in *Advances in Cryptology - Eurocrypt '90*, ser. Lecture Notes in Computer Science, I. B. Damgård, Ed., vol. 473. Springer-Verlag, 1990, pp. 374–388.

[3] W. Soo, A. Samsudin, and A. Goh, "Efficient mental card shuffling via optimised arbitrary-sized benes permutation network," in *Information Security Conference*, ser. Lecture Notes in Computer Science, vol. 2433. Springer-Verlag, 2002, pp. 446–458.

[4] J. Castellà-Roca, J. Domingo-Ferrer, A. Riera, and J. Borrell, "Practical mental poker without a ttp based on homomorphic encryption," in *Progress in Cryptology-Indocrypt'2003*, ser. Lecture Notes in Computer Science, T. Johansson and S. Maitra, Eds., no. 2904. Berlin: Springer-Verlag, 2003, pp. 280–294.

[5] A. Barnett and N. Smart, "Mental poker revisited," in *Proc. Cryptography and Coding*, ser. Lecture Notes in Computer Science, vol. 2898. Springer-Verlag, December 2003, pp. 370–383.

[6] R. Rivest, L. Adleman, and M. Dertouzos, "On data banks and privacy homomorphisms," in *Foundations of Secure Computation*, R. DeMillo, Ed. New York: Academic Press, 1978, pp. 169–179.

[7] T. ElGamal, "A public-key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, pp. 469–472, July 1985.

[8] J. Domingo-Ferrer, "A new privacy homomorphism and applications," *Information Processing Letters*, vol. 60, no. 5, pp. 277–282, December 1996. [Online]. Available: citeseer.nj.nec.com/290190.html

[9] J. Domingo-Ferrer, "A provably secure additive and multiplicative privacy homomorphishm," in *Information Security*, ser. Lecture Notes in Computer Science, A. Chan and V. Gligor, Eds., vol. 2433. Springer Verlag, 2002, pp. pp. 471–483.

[10] D.-R. Denning, *Cryptography and Data Security*, 2nd ed. Addison-Wesley, January 1983.

[11] C. Crépeau, "A secure poker protocol that minimizes the effect of player coalitions," in *Advances in Cryptology - Crypto '85*, ser. Lecture Notes in Computer Science, H. C. Williams, Ed., vol. 218. Berlin: Springer-Verlag, 1985, pp. 73–86.

[12] M. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal of Computing*, vol. 18, no. 1, pp. 186–208, 1989.

[13] E. Karnin, J. Green, and M. Hellman, "On secret sharing systems," *IEEE Trans. Information Theory*, vol. IT-29, no. 1, pp. 35–41, 1983.

[14] A. Shamir, "How to share a secret," in *Communications of the ACM*, vol. 22, 1979, pp. 612–613.

[15] M. Jakobsson and A. Juels, "Millimix: Mixing in small batches," DIMACS Technical Report, Tech. Rep. 99-33, 1999.

[16] P. Paillier, "Public key cryptosystems based on composite residue classes," in *Advances in Cryptology – EuroCrypt'99*, ser. Lecture Notes in Computer Science, vol. 1592. Springer-Verlag, 1999, pp. 223–238.

[17] G. Brassard, C. Crépeau, and J. Robert, "All-or-nothing disclosure of secrets," in *Advances in Cryptology Crypto'86*, ser. Lecture Notes in Computer Science, vol. 263. Springer-Verlag, 1986, pp. 234–238.