Secure cloud computing algorithms for discrete constrained potential games

Yang Lu *, Minghui Zhu *

* Department of Electrical Engineering, Pennsylvania State University, 201 Old Main, University Park, PA, 16802 USA (e-mail: yml5046@psu.edu (Y. Lu), muz16@psu.edu (M. Zhu)).

Abstract: In this paper, we study secure cloud computing problem for a class of discrete constrained potential games. In the games, certain functions are confidential for the system operator and not disclosed to any other participant. Meanwhile, each agent is unwilling to disclose its private functions and states to any other participant. By utilizing reinforcement learning and homomorphic encryption, we propose a distributed algorithm where (i) both the confidentiality for the system operator and the privacy for the agents are protected; (ii) the convergence to Nash equilibria is formally ensured.

© 2015, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.

Keywords: Secure computation, cloud computing, potential games, reinforcement learning, homomorphic encryption.

1. INTRODUCTION

Distributed computation has been widely applied to control multi-agent networks due to a number of advantages; i.e., scalability and robustness. In distributed computation, distributed data sharing among the participants is necessary to achieve network-wide goals. However, it also causes the concern that private and confidential information of legitimate entities could be leaked to unauthorized entities. Such concern demands for new distributed algorithms that can accomplish the tasks of distributed computation and simultaneously protect the information security of legitimate entities.

Contribution. This paper studies the secure cloud computing problem for a class of discrete constrained potential games. In the games of interest, there are certain confidential functions which the system operator is unwilling to disclose. In addition, each agent is unwilling to disclose its private functions and states to any others. We propose a distributed algorithm which integrates reinforcement learning [Zhu and Martínez, 2013] and homomorphic encryption [Sen, 2013]. At each iteration, each agent, on one hand, reinforces the successful state in the most recent history, and on the other hand, randomly explores feasible states. The decision making is only based on the realized values of its private objective function and the realized values are securely computed by using a new homomorphic encryption based polynomial computation algorithm. First, the system operator computes each multiplication term of the polynomial by operating on the data encrypted by the agents, who then decrypt the computing results to obtain the true values of the terms. After that, together with a third party, by using a non-singular masking matrix, all the participants collectively perform an inner product operation to obtain the true value of the polynomial.

We show that the proposed algorithm is able to protect both the confidentiality for the system operator and the privacy for the agents and ensure the convergence of Nash equilibrium with probability one. Due to the space limit, we omit most analysis in this paper. The complete results of the paper are included in the technical report [Lu and Zhu, 2015].

Literature review:

This paper studies a class of potential games. Potential games were first proposed in [Monderer and Shapley, 1996]. Several types of potential games have been studied, e.g., ordinal potential games, cardinal potential games and pseudo-potential games [Kukushkin, 1999], [Jehiel et al., 2004], [Dubey et al., 2006] and different algorithms for potential games have been developed such as best response, better response and MaxLogit [Voorneveld, 2000], [Song et al., 2011], [Canzogian et al., 2013]. Recently, potential games have been broadly applied to cooperative control, e.g., [Marden et al., 2009], [Zhu, 2014].

Another set of relevant work is information security. There have been different secure computing techniques, e.g., differential privacy [Dwork et al., 2006], secure multiparty computation (SMC) [Shamir, 1979] and homomorphic encryption [Sen, 2013]. Due to the probabilistic nature, differential privacy cannot guarantee perfect correctness of computation. In existing SMC literature, the parties collectively compute a function, where it requires that all the parties know the structure of the function. Homomorphic encryption has been used in secure cloud computing, e.g., [López-Alt et al., 2012]. However, current work on homomorphic encryption only considers the privacy for the agents but does not take into account the confidentiality for the system operator.

The above short analysis indicates that it still lacks algorithmic frameworks to solve games in a secure manner.
Notations: In this paper, the following notations will be used.

- $0_n$: zero vector of size $n$.
- $\mathbb{R}_+^n$: set of non-negative real column vectors of size $n$.
- $[x]^T$: projection of vector $x = [x_1, \ldots, x_n]^T$ onto the non-negative space, i.e., $[x]^+ = [x_1^+, \ldots, x_n^+]^T$ where for each $i = 1, \ldots, n$, $x_i = x_i$ if $x_i \geq 0$ and $\bar{x}_i = 0$ if $x_i < 0$.
- $|x|$: entry-wise absolute operation on vector $x$, i.e., $|x| = [|x_1|, \ldots, |x_n|]^T$.
- $x^a$: entry-wise exponent operation with exponent $a$, i.e., $x^a = [x_1^a, \ldots, x_n^a]^T$.
- $\mathbb{P}(E)$: probability that an event $E$ happens.
- $\mathbb{P}_Z$: projection operator onto a convex and compact set $Z \subseteq \mathbb{R}^n$, i.e., for $z \in \mathbb{R}^n$, $\mathbb{P}_Z[z] = \arg\min_{x \in Z} \|x - z\|$.

Given a non-negative scalar sequence $(\alpha(k))_{k \geq 0}$, it is summable if $\sum_{k=0}^{\infty} \alpha(k) < +\infty$ and square summable if $\sum_{k=0}^{\infty} \alpha(k)^2 < +\infty$.

2. PROBLEM STATEMENT

In this section, we formalize the class of discrete constrained potential games and the security issues concerned in this paper.

2.1 Problem formulation

Consider a set of agents $\mathcal{V} = \{1, \ldots, N\}$, a system operator (SO) and a third party (TP). The interconnections among them are represented by an undirected communication graph $\mathcal{G} = (\mathcal{V} \cup \{\text{SO}, \text{TP}\}, \mathcal{E})$ where $\mathcal{E}$ is the set of edges. If there is an undirected communication channel between $i$ and $j$, where $i, j \in \mathcal{V} \cup \{\text{SO}, \text{TP}\}$ and $i \neq j$, then $(i, j), (j, i) \in \mathcal{E}$. Since $\mathcal{G}$ is undirected, we consider $(i, j)$ and $(j, i)$ as the same channel.

Assumption 2.1. In $\mathcal{G}$, (SO, TP) $\in \mathcal{E}$. For each $i \in \mathcal{V}$, (SO, i), (TP, i) $\in \mathcal{E}$. For any pair $i, j \in \mathcal{V}$, $(i, j) \notin \mathcal{E}$.

Figure 1 illustrates the interconnections between $\mathcal{V} \cup \{\text{SO}, \text{TP}\}$. The solid lines represent the communication channels. Many networked engineering systems operate in hierarchical structures as Figure 1, e.g., Internet, power grid and transportation systems, where an SO is placed at the top layer and agents are placed at the bottom layer [Wu et al., 2005]. Assumption 2.1 is widely used in, e.g., network flow control [Low and Lapsley, 1999]. The operation conducted by the SO can be viewed as cloud computing, where the SO is the cloud provider while the agents are remote data sources. The TP is introduced due to the security concern, as will be illustrated later. We point out that the TP is untrustful and could be malicious.

Each agent $i \in \mathcal{V}$ has a state $x[i] = [x[i]^1, \ldots, x[i]^n]^T \in \mathbb{R}_+^n$. Denote by $x = [x^1^T, \ldots, x^N_T]^T \in \mathbb{R}_+^n$ the overall vector of all agents’ states where $n = \sum_{i=1}^N n_i$, and by $X = \prod_{i=1}^N \mathbb{R}_+^n$ the space of $x$. For each $i \in \mathcal{V}$, denote by $x[i] = [x[i]^1, \ldots, x[i]^n]^T \in \mathbb{R}_+^n$. The vector of all the other agents’ states except that of agent $i$, and by $X_{-i} = \prod_{j \neq i} X_j \subseteq \mathbb{R}_+^{n-n_i}$, the space of $x[i]$.

Given the joint state $x[i\sim]$, each agent $i \in \mathcal{V}$ aims to solve the following constrained optimization problem:

$$\min_{x[i] \in X} f_i(x[i], x[i\sim])$$
$$\text{s.t. } g_i^0(x[i], x[i\sim]) \leq p_i, \quad h_i^0(x[i], x[i\sim]) = 0,$$

where $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $g_i^0 : \mathbb{R}^n \rightarrow \mathbb{R}^{p_i}$ and $h_i^0 : \mathbb{R}^n \rightarrow \mathbb{R}^{q_i}$. For each $i \in \mathcal{V}$, denote $g_i = [g_i^0, \ldots, g_i^{q_i}]^T$ with $g_i^\ell : \mathbb{R}^n \rightarrow \mathbb{R}$ for $\ell = 1, \ldots, p_i$, and $h_i = [h_i^{0}, \ldots, h_i^{q_i}]^T$ with $h_i^\ell : \mathbb{R}^n \rightarrow \mathbb{R}$ for $\ell = 1, \ldots, q_i$.

Assumption 2.2. For all $i \in \mathcal{V}$, $X_i$ includes a finite number of states, $p_i = p$ and $q_i = q$.

Assumption 2.3. There exist functions $F : \mathbb{R}^n \rightarrow \mathbb{R}$, $G : \mathbb{R}^n \rightarrow \mathbb{R}^p$ and $H : \mathbb{R}^n \rightarrow \mathbb{R}^q$ such that, for any $i \in \mathcal{V}$, any $x[i\sim] \in X_{-i}$ and any $x[i]^\sim \in X_i$, it holds that $f_i(x[i], x[i\sim]) = f_i(x[i], x[i\sim]) - f_i(x[i], x[i\sim]) - F(x[i], x[i\sim]) - F(x[i], x[i\sim])$, $g_i^0(x[i], x[i\sim]) - g_i^0(x[i], x[i\sim]) - G(x[i], x[i\sim]) - G(x[i], x[i\sim])$ and $h_i^0(x[i], x[i\sim]) - h_i^0(x[i], x[i\sim]) = H(x[i], x[i\sim]) - H(x[i], x[i\sim])$.

By Assumptions 2.2 and 2.3, the collection of the coupled optimization problems (1) consists of a discrete constrained potential (DCP) game. The definition of Nash equilibrium (NE) of the DCP game is given as follows.

Definition 2.1. An NE of the DCP game is a state $x \in X$ such that for all $i \in \mathcal{V}$, $g_i^0(x[i], x[i\sim]) \leq 0_p$, $h_i^0(x[i], x[i\sim]) = 0_q$, and for all $x[i]^\sim \in X_i$ such that $g_i^0(x[i], x[i\sim]) \leq 0_p$, $h_i^0(x[i], x[i\sim]) = 0_q$, it holds that $f_i(x[i]) \leq f_i(x[i], x[i\sim])$.

Denote by $X_{DCP}$ the set of NEs of the DCP game.

2.2 Confidentiality and privacy issues

The distributed NE computation of the DCP game requires distributed data sharing between the participants which causes concerns of information leakage. This motivates us to study the security issue in cloud computing. In this paper, we are concerned with semi-honest adversaries and information-theoretic security. A participant is semi-honest if it correctly follows the algorithm but attempts to use the received messages to infer private information of legitimate participants ([Hazay and Lindell, 2010], pp. 20). A cryptosystem is information-theoretically secure if its security derives purely from information theory, i.e., if an adversary does not have enough information, then it cannot break the system even if it has unlimited computing power [Shannon, 1949]. Assume that in $\mathcal{G}$, each $(i, j) \in \mathcal{E}$ is secure, that is, only $i$ and $j$ can access the message sent via $(i, j)$ and no other participant can access the message.
In this paper, we consider both the confidentiality issue for the SO and the privacy issue for the agents.

**Confidentiality.** For each agent \( i \in \mathcal{V} \), there are \( r_i \) subfunctions in \( f_i \), \( g_i \) and \( h_i \) which are private pieces of information of the SO and the SO is unwilling to disclose the structures of these subfunctions to \( \mathcal{V} \cup \{ \text{TP} \} \). This is the confidentiality issue for the SO. Denote by \( \mathcal{F}_i^T = [F_i^T, \ldots, F_i^{|\mathcal{N}|T}]^T : \mathbb{R}^n \rightarrow \mathbb{R}^c \) the vector of such subfunctions in agent \( i \)'s optimization problem defined by (1). Given any state \( x \), the SO is willing to publicize the value of \( \mathcal{F}_i(x) \) to agent \( i \). Denote by \( \mathcal{F} = [\mathcal{F}_1^T, \ldots, \mathcal{F}_|\mathcal{N}|^T]^T \) the overall confidential functions for the SO.

**Privacy.** Each agent \( i \) is not willing to share the structures of its cost function \( f_i \), constraints \( g_i \) and \( h_i \) and the value of its state \( x_i \) with \( \mathcal{V} \setminus \{ i \} \cup \{ \text{SO}, \text{TP} \} \). This is the privacy issue for the agents. For each \( i \in \mathcal{V} \), denote \( \mathcal{W}_i^T = [f_i, g_i^T, h_i^T]^T : \mathbb{R}^n \rightarrow \mathbb{R}^{c+p} \).

**Assumption 2.4.** For each \( i \in \mathcal{V} \): 1) For \( \ell = 1, \ldots, r_i \), \( \mathcal{F}_i^\ell \) is a scalar polynomial function of \( x \); 2) There exists a function \( \mathcal{W}_i^\ell : \mathbb{R}^{n} \times \mathbb{R}^{r_i} \rightarrow \mathbb{R}^{1+p+q} \) such that \( \mathcal{W}_i^\ell(x) = \mathcal{W}_i^\ell(x_i, \mathcal{F}_i^\ell(x)) \) and agent \( i \) knows the structure of \( \mathcal{W}_i^\ell \).

By Assumption 2.4, the confidentiality and privacy issues concerned in this paper reduce to the problem of secure polynomial computation. The polynomial assumption is valid in many applications, e.g., the sum-of-squares (SOS) optimization, where the cost function is linear and the constraints are polynomials with SOS property [Prajna et al., 2004], and the model predictive control, where the optimization problem therein usually adopts a quadratic cost function with linear constraints [Lu et al., 2014].

### 2.3 Objective

The objective of this paper is to design a distributed cloud computing algorithm to solve the DCP game such that the convergence of NE is ensured and both the confidentiality for the SO and the privacy for the agents are protected.

### 3. SECURE POLYNOMIAL COMPUTATION ALGORITHM

We pointed out in the last section that, by Assumption 2.4, the security issues concerned in this paper reduce to the problem of secure polynomial computation. In this section, we propose a homomorphic encryption based secure polynomial computation algorithm. First, we illustrate some preliminaries for homomorphic encryption. After that, we present the algorithm and its analysis.

#### 3.1 Preliminaries for homomorphic encryption

Homomorphic encryption is a form of encryption that allows computations to be carried out on ciphertext, thus generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext. Based on different encryption methods, a homomorphic encryption scheme can be classified as deterministic or probabilistic. Roughly speaking, if some random parameters are used in the encryption operation, then the scheme is probabilistic, otherwise, it is deterministic. In this paper, because we want to achieve perfect computation correctness, we are concerned with deterministic homomorphic encryption scheme which is defined as follows.

**Definition 3.1.** Let \( \mathcal{S} \) be the message space and \( \odot \) be a binary operation. A deterministic homomorphic encryption scheme on \( (\mathcal{S}, \odot) \) is a quadruple \((K, E, D, T)\), where \( K \) is the key-generation algorithm, \( E \) is the encryption algorithm, \( D \) is the decryption algorithm and \( T \) is a binary operation with desired homomorphic property, such that, for any key \( k \) generated by \( K \), for any \( c, c' \in \mathcal{S} \), it holds that \( D(T(E(c, k_e), E(c', k_d)), k_e) = c \odot c' \).

The intuition of the above definition is that homomorphic encryption allows computations to be carried out on ciphertext, thus generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext. In this paper, we use the multiplicatively homomorphic encryption scheme, that is, \( \odot \) in Definition 3.1 is the multiplication operation. There are several classical multiplicatively homomorphic encryption schemes such as RSA (named by the initials of the surnames of the inventors, Ron Rivest, Adi Shamir and Leonard Adleman) and ElGamal. In this paper, we choose RSA [Rivest et al., 1978] as our implementing scheme. In RSA, the key \( k_e \) is a positive scalar \( a \in \mathbb{R}_{++}^+ \). The encryption operation of a message \( c \in \mathbb{R} \) is \( E(c, a) = c^a \). The decryption operation of an encrypted value \( d \in \mathbb{R} \) is \( D(d, a) = d^a \). The binary operation is the multiplication operation. We then have, for any \( c, c' \in \mathbb{R} \), \( D(T(E(c, a), E(c', a)), a) = cc' \). In this way, if two agents send their encrypted inputs \( c^a \) and \( c'^a \) to the SO and ask the SO to compute and send back the product of the two encrypted inputs, \( (cc')^a \), they can decrypt \( (cc')^a \) and obtain the desired product \( cc' \). During this process, since the SO does not know the value of \( a \), it cannot infer the values of \( c \) or \( c' \).

#### 3.2 Secure polynomial computation algorithm

For convenience of notation, for a polynomial of \( x \), we call each term without the coefficient a multiplication term. For example, for the polynomial \( 6x_1^2 + 4x_2^3 + 5 \), \( x_1^2 \), \( x_2^3 \), and 1 are multiplication terms with coefficients 6, 4 and 5, respectively. For \( i \in \mathcal{V}, \) for \( j = 1, \ldots, r_i \), let \( \mathcal{F}_i^j(x) = \sum_{\ell=1}^{m_i,j} C_i^j[\ell]^\ell Q_i[\ell]^\ell \), where for \( \ell = 1, \ldots, m_i,j \), \( Q_i[\ell]^\ell \) is a multiplication term with coefficient \( C_i^j[\ell] \). Denote \( Q_i^\ell = [Q_i^\ell, \ldots, Q_i^{m_i,j}]^T \) and \( C_i^\ell = [C_i^\ell, \ldots, C_i^{m_i,j}]^T \).

Before running Algorithm 1, all the agents agree on a RSA key \( a \in \mathbb{R}_{++} \) which is unknown to \( \{ \text{SO}, \text{TP} \} \). By running \( \text{OUT}[^1], \ldots, \text{OUT}[\mathcal{N}] \) = alg3\( \text{IN}[^1], \ldots, \text{IN}[\mathcal{N}] \), the algorithm takes \( \text{IN}[^i] \) as the input for agent \( i \) and outputs \( \text{OUT}[i] \) to agent \( i \). Denote \( \text{IN} = [\text{IN}[^1]^T, \ldots, \text{IN}[\mathcal{N}]^T]^T \).

To help understand the steps of Algorithm 1, we first provide a simple example for concreteness. Consider the polynomial \( 6x_1^2 + 4x_2^3 + 5 \), where \( x_1^2 \) and \( x_2^3 \) are scalars. First, for \( i = 1, 2, 3 \), each agent \( i \) sends \( (x_i^2)^a \) to the SO. The SO computes the encrypted multiplication terms \( (x_i^2)^2a \) and \( (x_i^2)^3a \). Notice that the coefficients are not included in the computation at this step. This is because that the RSA cannot deal with multiplication...
by coefficients. We will deal with the multiplication by coefficients and the addition of the terms by using the TP and a masking matrix. The SO generates an arbitrary non-singular 3-by-3 masking matrix $B$, say, $B_1 = [1, 2, 3]^T$, $B_2 = [2, 1, 0]^T$ and $B_3 = [3, 0, 1]^T$, where $B_i$ is the $i$-th column of $B$ for $i = 1, 2, 3$. The SO then sends $((x_1^1 x_2^2)^a, B_1)$ to agent 3, $((x_1^3 x_2^3)^a, B_2)$ to agent 2 and $(1, B_3)$ to agent 1, respectively (the way of distributing the encrypted multiplications terms is not unique). Agent 1 decrypts 1, which is still 1, and sends $B_3$ to the TP. Agent 2 decrypts $((x_1^1 x_2^2)^a)$ to obtain $x_1^1 x_2^2$ and sends $x_1^1 x_2^2 B_2$ to the TP. Agent 3 decrypts $((x_1^3 x_2^3)^a)$ to obtain $x_1^3 x_2^3$ and sends $x_1^3 x_2^3 B_1$ to the TP. The SO sends $[6, 4, 5] B^{-1}$ to the TP. The TP then computes $[6, 4, 5] B^{-1} (x_1^1 x_2^2 B_1 + x_1^3 x_2^3 B_2 + B_3) = [6, 4, 5] B^{-1} B [x_1^1 x_2^2, x_1^3 x_2^3, 1]^T = [6, 4, 5] [x_1^1 x_2^2, x_1^3 x_2^3, 1]^T = 6x_1^1 x_2^2 + 4x_1^3 x_2^3 + 5$, which is the desired polynomial. Notice that the step of multiplication by coefficients and the addition of the terms is an inner product operation, that is, for the example here, to compute $[6, 4, 5] [x_1^1 x_2^2, x_1^3 x_2^3, 1]^T$. This inner product operation cannot be performed by the SO, because otherwise the SO has to know the true values of $x_1^1 x_2^2$ and $x_1^3 x_2^3$. Since the SO also knows the encrypted values $(x_1^1 x_2^2)^a$ and $(x_1^3 x_2^3)^a$, it then be able to infer the value of $a$ and further infer the values of $x_1^1$, $x_2^2$ and $x_3$. The inner product operation cannot be performed by any of the agents either, since otherwise the agent has to know the coefficients of the polynomial. Therefore, to protect both the confidentiality for the SO and the privacy for the agents, we introduce the TP to perform the inner product operation. The usage of the non-singular matrix $B$ is to mask the values of the multiplication terms and the coefficients from the TP. Since the multiplication terms sent to the TP are multiplied by $B$ and the coefficients sent to the TP are multiplied by $B^{-1}$, and the TP does not know $B$ or $B^{-1}$, it cannot infer the values of the multiplication terms or the coefficients. When the TP performs the multiplication operation, $B$ and $B^{-1}$ cancels and the value of the desired polynomial is correctly computed.

The general steps of Algorithm 1 are informally stated as follows. First, each agent $i \in \mathcal{V}$ encrypts its input by RSA, i.e., $s_{\text{IN}}^i = [\mathcal{IN}]^a_i$, and sends $s_{\text{IN}}^i$ to the SO. Then, for each agent $i$, there are $r_i$ rounds such that, at each round $j$, $\mathcal{OUT}_j^i = \mathcal{F}_{j,i}^\uparrow(\mathcal{IN})$ is computed and sent to agent $i$. For each agent $i \in \mathcal{V}$, at round $j$, the SO computes each encrypted version of $Q_{j,i}^\uparrow$ by treating $s_{\text{IN}}^i$ as $\mathcal{IN}$ for $\ell \in \mathcal{V}$. That is, e.g., if $Q_{j,i}^\uparrow = (x_1^1)^2 x_2^2$ then the SO computes $Q_{j,i}^\uparrow = ([\mathcal{IN}]^i)^2 \mathcal{IN}_2^2$. The SO then generates a non-singular $m_{i,j}$-by-$m_{i,j}$ matrix $B_{j,i}^\downarrow = [B_{j,i,1}^\downarrow, \ldots, B_{j,i,m_{i,j}}^\downarrow]$, where $B_{j,i}^\downarrow$ is the $\ell$-th column of $B_{j,i}$. After that, the SO sends each pair $(Q_{j,i}^\uparrow, B_{j,i}^\downarrow)$ to the agents in the way described in the algorithm. The specific way of data sending at this step is to ensure that no individual agent can infer the structure of the processed polynomial, as will be shown in the following analysis part. Then, each agent $i \in \mathcal{V}$ decrypts each $Q_{j,i}^\uparrow$ and computes $Y_{j,i}^\uparrow = B_{j,i}^\downarrow Q_{j,i}^\uparrow$ and sends it to the TP, while the SO sends $Z_{j,i}^\uparrow = B_{j,i}^\downarrow T - 1$ to the TP. The TP then computes $\mathcal{OUT}_j^i = Z_{j,i}^\uparrow \sum_{\ell=1}^{m_{i,j}} Y_{j,i}^\uparrow$ and sends it to agent $i$. Here, the matrix $B_{j,i}^\uparrow$ has the masking function, i.e., it masks the values of $Q_{j,i}^\uparrow$ and $C_{j,i}^\uparrow$ from the TP. We will show in the following analysis part that $\mathcal{OUT}_j^i = \mathcal{F}_{j,i}^\uparrow(\mathcal{IN})$. After all the $r_i$ rounds, agent $i$ forms the vector $\mathcal{OUT}_i^\uparrow = [\mathcal{OUT}_{1,i}^\uparrow, \ldots, \mathcal{OUT}_{r_i,i}^\uparrow]^T = \mathcal{F}_{j,i}^\uparrow(\mathcal{IN})$.

**Algorithm 1: Secure polynomial computation algorithm**

Syntax: $(\mathcal{OUT}_1^\uparrow, \ldots, \mathcal{OUT}_N^\uparrow) = \text{alg3}([\mathcal{IN}]^1, \ldots, [\mathcal{IN}]^N)$

for $i \in \mathcal{V}$ do

Agent $i$ sends $s_{\text{IN}}^i = [\mathcal{IN}]^i a = [\mathcal{IN}]^i a$ to the SO;

for $i \in \mathcal{V}$ do

for $j = 1, \ldots, r_i$ do

The SO computes each encrypted $Q_{j,i}^\uparrow$, denoted by $\tilde{Q}_{j,i}^\uparrow$, for $\ell = 1, \ldots, m_{i,j}$, by treating $s_{\text{IN}}^i$ as $\mathcal{IN}$ for $\ell \in \mathcal{V}$;

The SO generates a non-singular $m_{i,j}$-by-$m_{i,j}$ matrix $B_{j,i}^\downarrow = [B_{j,i,1}^\downarrow, \ldots, B_{j,i,m_{i,j}}^\downarrow]$;

The SO sends $(\tilde{Q}_{j,i}^\uparrow, B_{j,i}^\downarrow)$ for $\ell = 1, \ldots, m_{i,j}$ to the agents such that, each pair is sent to only one agent, not all the pairs are sent to a single agent, and if some $\tilde{Q}_{j,i}^\uparrow$ only includes the data of a single agent $l \in \mathcal{V}$, then this pair is not sent to agent $l$;

for $\ell \in \mathcal{V}$ do

Agent $l$ decrypts each $\tilde{Q}_{j,i}^\uparrow$ it holds by computing $\tilde{Q}_{j,i}^\uparrow = B_{j,i}^\downarrow Q_{j,i}^\uparrow$ for each $Q_{j,i}^\uparrow$ it holds and sends each $Y_{j,i}^\uparrow$ to the TP;

The SO sends $Z_{j,i}^\uparrow = B_{j,i}^\downarrow T - 1$ to the TP;

The TP sends $\mathcal{OUT}_j^i = Z_{j,i}^\uparrow \sum_{\ell=1}^{m_{i,j}} Y_{j,i}^\uparrow$ to agent $i$;

Agent $i$ forms $\mathcal{OUT}_i^\uparrow = [\mathcal{OUT}_{1,i}^\uparrow, \ldots, \mathcal{OUT}_{r_i,i}^\uparrow]^T$;


3.3 Analysis

**Theorem 3.1.** Under Assumptions 2.1 and 2.4, by Algorithm 1, the following claims hold:

1) Correctness: For each $i \in \mathcal{V}$, $\mathcal{OUT}_i^\uparrow = \mathcal{F}_{j,i}^\uparrow(\mathcal{IN})$.

2) Confidentiality: For the SO, the structure of $\mathcal{F}$ is unknown to $\mathcal{V} \setminus \{\mathcal{TP}\}$.

3) Privacy: For each agent $i$, the value of its input $[\mathcal{IN}]^i$ is unknown to $(\mathcal{V} \setminus \{i\}) \cup \{\text{SO}, \text{TP}\}$.

**Remark 3.1.** In existing secure cloud computing literature, only the privacy for the agents is considered. Algorithm 1 is able to protect both the confidentiality for the SO and the privacy for the agents.

Now we analyze the online computational complexity of computing one polynomial by Algorithm 1. We choose $\mathcal{F}_{j,i}^\uparrow$ for concreteness, where $i \in \mathcal{V}$ and $j \in \{1, \ldots, r_i\}$. Assume that in $\mathcal{F}_{j,i}^\uparrow$, there are totally $o_j^i$ times of multiplications. The generation of $B_{j,i}^\downarrow$ and the computation of $Z_{j,i}^\uparrow$ can be done off-line by the SO beforehand, thus, we do not count these operations into the online computational complexity. To compute $\mathcal{F}_{j,i}^\uparrow$, it collectively takes the participants
2N times of exponent computations, $e_{ij}^{[2]} + m_{ij}$ times of multiplications, $m_{ij}^2 + 2m_{ij}$ - 2 times of additions and $N + 2m_{ij}$ + 2 times of communications.

4. DISCRETE POTENTIAL GAMES

In this section, by integrating reinforcement learning and homomorphic encryption, we propose a distributed secure cloud computing algorithm to solve the DCP game.

4.1 Game transformation

To relax the constraints, we adopt the approach of penalty functions. For each $i \in \mathcal{V}$, define the local penalty function $\mathcal{P}_i : \mathbb{R}^n \times \mathbb{R}_+^n \times \mathbb{R}_+^q \rightarrow \mathbb{R}$ by $\mathcal{P}_i (x, \mu, \lambda) \triangleq f_i (x) + \mu^T [g_i^T (x)]^+ + \lambda^T [h_i^T (x)],$ where $\mu = [\mu_1, \cdots, \mu_p]^T \in \mathbb{R}_+^p$ and $\lambda = [\lambda_1, \cdots, \lambda_q]^T \in \mathbb{R}_+^q$. Fix $\mu$ and $\lambda$. Then, it can be checked that $\mathcal{H} (x, \mu, \lambda) = \mathcal{F} (x) + \mu^T G (x) + \lambda^T H (x)$ is a potential function such that for any $i \in \mathcal{V}$, any $x^{[-i]} \in X_{-i}$ and any $x^i, \bar{x}^i \in X_i$, it holds that $\mathcal{P}_i (x^i, x^{[-i]}, \mu, \lambda) - \mathcal{P}_i (\bar{x}^i, x^{[-i]}, \mu, \lambda) = \mathcal{H} (x^i, x^{[-i]}, \mu, \lambda) - \mathcal{H} (\bar{x}^i, x^{[-i]}, \mu, \lambda)$.

For each $i \in \mathcal{V}$, consider the following discrete unconstrained optimization problem parameterized by $\mu$ and $\lambda$: \begin{equation}
\min_{x^i \in X} \mathcal{P}_i (x^i, x^{[-i]}, \mu, \lambda).
\end{equation}

Due to the existence of potential function $\mathcal{H}$, the collection of the coupled optimization problems (2) consists of a discrete unconstrained potential game parameterized by $\mu$ and $\lambda$ (DUP($\mu, \lambda$) game). The definition of NE of the DUP($\mu, \lambda$) game is presented as follows.

**Definition 4.1.** A joint state $\bar{x} \in X$ is an NE of the DUP($\mu, \lambda$) game if it holds that $\mathcal{P}_i (\bar{x}, \mu, \lambda) \leq \mathcal{P}_i (x^i, \bar{x}^{[-i]}, \mu, \lambda), \forall x^i \in X_i, \forall i \in \mathcal{V}$.

Denote by $\mathcal{X}_{\text{DUP}} (\mu, \lambda)$ the set of NEs of the DUP($\mu, \lambda$) game. Since the DUP($\mu, \lambda$) game is a potential game, $\mathcal{X}_{\text{DUP}} (\mu, \lambda) \neq \emptyset$. We now proceed to study the relations between the DCP game and the DUP($\mu, \lambda$) game. Before that, we have the following assumption.

**Assumption 4.1.** For any infeasible $\bar{x} \in X$, there exists $i \in \mathcal{V}$ for which $\bar{x} \not\in \{ x \in X | g_i^T (x) \leq 0 \}, h_i (x) = 0 \}$ and agent $i$ can unilaterally deviate from $\bar{x}$ to another state $\tilde{x} \in X_i$ such that $[g_i (\tilde{x})] \leq [g_i (\bar{x})] + \mu_i$ for all $i = 1, \cdots, p$, $h_i (\tilde{x}) \leq h_i (\bar{x})$ for all $i = 1, \cdots, q$, and (i) there exists at least one $j \in \{1, \cdots, q\}$ such that $h_j (\bar{x}) \neq 0$ and $h_j (\tilde{x}) < h_j (\bar{x})$; (ii) or there exists at least one $j \in \{1, \cdots, q\}$ such that $h_j (\bar{x}) = 0$ and $h_j (\tilde{x}) < h_j (\bar{x})$; (iii) or both.

For convenience of notation, denote: 

$$
\sigma_{\text{max}} = \max_{i \in \mathcal{V}} \max_{x^i \in X_i} \{ f_i (x^i, x^{[-i]}), f_i (x^i, x^{[-i]}) \},
$$

$$
\sigma_{\mu, \text{min}} = \min_{i \in \mathcal{V}} \min_{x^{[-i]} \in X} \{ [g_i (x^{[-i]}) - g_i (\bar{x}^{[-i]})]^+, [g_i (\bar{x}^{[-i]}), x^{[-i]}]^+] \},
$$

$$
\sigma_{\lambda, \text{min}} = \min_{i \in \mathcal{V}} \min_{x^{[-i]} \in X} \{ [h_i (x^{[-i]}) - h_i (\bar{x}^{[-i]})]^+, [h_i (\bar{x}^{[-i]}), x^{[-i]}]^+] \},
$$

for $\ell = 1, \cdots, p$, where $\Theta_{\text{e}} = \{ x^i, x^{[-i]} \in X_i, x^{[-i]} \in X_{-i}, \exists \tilde{t} \in \mathcal{E}_{\text{e}} \}$ and $\Theta_{\text{t}} = \{ x^i, x^{[-i]} \in X_i, x^{[-i]} \in X_{-i}, \exists \tilde{t} \in \mathcal{E}_{\text{t}} \}$.

**Lemma 4.1.** If $\mu_\ell > \frac{\sigma_{\text{max}}}{\sigma_{\mu, \text{min}}} \text{ for all } \ell = 1, \cdots, p$ and $\lambda_\ell > \frac{\sigma_{\text{max}}}{\sigma_{\lambda, \text{min}}} \text{ for all } \ell = 1, \cdots, q$, then any NE of the DUP($\mu, \lambda$) game is an NE of the DCP game.

By Assumption 2.4, for each $i \in \mathcal{V}$, there exists a function $\mathcal{P}_i : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}_+^n \times \mathbb{R}_+^q \rightarrow \mathbb{R}$ such that $\mathcal{P}_i (x, \mu, \lambda) = \mathcal{P}_i (x^i, \mathcal{F}_i (x), \mu, \lambda)$ and agent $i$ knows the structure of $\mathcal{P}_i$.

4.2 Distributed computation algorithm

In this subsection, we propose a distributed algorithm by which the participants collectively compute NE of the DCP game and the confidentiality for the SO and the privacy for the agents are both protected.

Due to the security issues, we cannot adopt the best or better-response methods which are classical ways to solve potential games. Because, to compute best or better responses, each agent needs to completely know its own objective function and other agents’ states [Monderer and Shapley, 1996]. We address this problem by using reinforcement learning and homomorphic encryption.

Before running Algorithm 2, the agents agree on $\mu, \lambda$ whose values are fixed during the algorithm. The way of choosing $\mu$ and $\lambda$ will be illustrated by Theorem 4.1 in the next subsection. Algorithm 2 is informally stated as follows. At each iteration $k$, each agent $i$ first computes $\sigma (k)$ whose value is small and diminishing as $k$ increases to infinity. Then, each agent $i$ chooses $\tau_i (k)$ to be the iteration number with the lower cost within the last two iterations. Here, $x^i (\tau_i (k))$ is the more successful action of agent $i$ in the last two iterations. Then, each agent $i$ uniformly picks any state from $X_i \setminus \{ x^i (\tau_i (k)) \}$ as the new state $x^i (k + 1)$ with probability $\epsilon (k)$, and with probability $1 - \epsilon (k)$, chooses the new state $x^i (k + 1)$ by reinforcing the more successful decision $x^i (\tau_i (k))$ in the last two iterations. Then, the participants collectively run Algorithm 1 which outputs $v^i (k + 1) = \mathcal{F}_i (x (k + 1))$ to agent $i$. After that, with $v^i (k + 1)$, each agent $i$ computes the cost $\mathcal{P}_i$ at the next iteration $k + 1$. Note that, since each agent $i$ does not know the structure of $\mathcal{F}_i$, it cannot directly compute $\mathcal{P}_i$. Also, due to the privacy issue, the agents cannot directly send their states to the SO and let the SO do the computation.

4.3 Analysis

Observe that $z (k) \triangleq (x (k - 1), x (k))$ in Algorithm 2 constitutes a Markov chain on the space $X \times X$. The following theorem states that the algorithm asymptotically converges in probability to the set $\mathcal{X}_{\text{DC}} \triangleq \{ (s, s) | s \in \mathcal{X}_{\text{DC}} \}$ and at the same time the confidentiality for the system operator and the privacy for the agents are protected.

**Theorem 4.1.** Under Assumptions 2.1, 2.3, 2.4 and 4.1, by choosing $\hat{\mu}_\ell > \frac{\sigma_{\text{max}}}{\sigma_{\mu, \text{min}}} \text{ for all } \ell = 1, \cdots, p$ and $\hat{\lambda}_\ell > \frac{\sigma_{\text{max}}}{\sigma_{\lambda, \text{min}}} \text{ for all } \ell = 1, \cdots, q$, by Algorithm 2, the following claims hold:

1) Correctness: $\lim_{k \rightarrow +\infty} \mathbb{P}(z (k) \in \mathcal{X}_{\text{DC}}) = 1$. 

Algorithm 2: Distributed secure cloud computing algorithm

for $i \in V$ do
  Agent $i$ uniformly picks $x_i^0(0) \in X_i$;
  The participants collectively run $(u_i^1(0), \ldots, u_i^{N_i}(0)) = \text{alg3}(x_i^0(0), \ldots, x_i^{N_i}(0))$;
for $i \in V$ do
  Agent $i$ computes $u_i(0) = \tilde{P}_i(x_i^0(0), u_i^0(0), \tilde{\mu}, \tilde{\lambda})$;
  Agent $i$ uniformly picks $x_i^1(1) \in X_i$;
  The participants collectively run $(u_i^1(1), \ldots, u_i^{N_i}(1)) = \text{alg3}(x_i^1(1), \ldots, x_i^{N_i}(1))$;
for $i \in V$ do
  Agent $i$ computes $u_i(1) = \tilde{P}_i(x_i^1(1), u_i^1(1), \tilde{\mu}, \tilde{\lambda})$;
  while $k \geq 1$ do
    for $i \in V$ do
      Agent $i$ computes $\epsilon(k) = k - \sum_{i=1}^{N_i} x_i^{k+1}$;
      Agent $i$ computes $\tau_i(k)$ as follows:
      if $u_i(k-1) \leq u_i(k-1)$ then
        $\tau_i(k) = k$;
      else
        $\tau_i(k) = k - 1$;
      Agent $i$ uniformly picks $x_i^{k+1} \in X_i \setminus \{x_i^0(\tau_i(k))\}$ w.p. $\epsilon(k)$
      and chooses $x_i^{k+1}$ w.p. $1 - \epsilon(k)$, and sets
      $x_i^{k+1} = x_i^{k+1}$;
      The participants collectively run $(u_i^1(k+1), \ldots, u_i^{N_i}(k+1)) = \text{alg3}(x_i^{1}(k+1), \ldots, x_i^{N_i}(k+1))$;
    for $i \in V$ do
      Agent $i$ computes $u_i(k+1) = \tilde{P}_i(x_i^{k+1}(1), u_i^{(k+1)}, \tilde{\mu}, \tilde{\lambda})$;
      $k = k + 1$;

2) Confidentiality: For the SO, the structure of $F$ is unknown to $V \cup \{TP\}$.

3) Privacy: For each agent $i$, the structures of $f_i$, $g_i^i$ and $h_i^i$ and the values of $x_i^i(k)$ at each iteration $k$ are unknown to $(V \setminus \{i\}) \cup \{SO, TP\}$.

5. CONCLUSION

We have proposed a secure cloud computing algorithm for discrete constrained potential games. The algorithm is able to protect both the confidentiality for the system operator and the privacy for the agents via homomorphic encryption, and its convergence is ensured by reinforcement learning. One future direction is to investigate secure computation for continuous convex games.

REFERENCES