Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 8952219, 16 pages
https://doi.org/10.1155/2021/8952219

Research Article

A Privacy-Preserving Reinforcement Learning Approach for Dynamic Treatment Regimes on Health Data

Xiaoqiang Sun[1,2] **Zhiwei Sun**[3] **Ting Wang**[3] **Jie Feng**[4,5] **Jiakai Wei**[6] **and Guangwu Hu**[1]

[1]*School of Computer, Shenzhen Institute of Information Technology, Shenzhen 518172, China*
[2]*Guangdong Key Laboratory of Intelligent Information Processing, College of Electronics and Information Engineering, Shenzhen University, Shenzhen 518060, China*
[3]*School of Artificial Intelligence, Shenzhen Polytechnic, Shenzhen 518055, China*
[4]*Guangzhou Institute of Technology, Xidian University, Guangzhou 510555, China*
[5]*Shaanxi Key Laboratory of Information Communication Network and Security, Xi'an University of Posts & Telecommunications, Xi'an, Shaanxi 710121, China*
[6]*Department of Neonatology, Xi'an Children's Hospital, Xi'an Jiaotong University, Xi'an 710003, China*

Correspondence should be addressed to Zhiwei Sun; smeker@szpt.edu.cn

Received 13 September 2021; Accepted 22 October 2021; Published 23 November 2021

Academic Editor: Celimuge Wu

Based on the clinical states of the patient, dynamic treatment regime technology can provide various therapeutic methods, which is helpful for medical treatment policymaking. Reinforcement learning is an important approach for developing this technology. In order to implement the reinforcement learning algorithm efficiently, the computation of health data is usually outsourced to the untrustworthy cloud server. However, it may leak, falsify, or delete private health data. Encryption is a common method for solving this problem. But the cloud server is difficult to calculate encrypted health data. In this paper, based on Cheon et al.'s approximate homomorphic encryption scheme, we first propose secure computation protocols for implementing comparison, maximum, exponentiation, and division. Next, we design a homomorphic reciprocal of square root protocol firstly, which only needs one approximate computation. Based on the proposed secure computation protocols, we design a secure asynchronous advantage actor-critic reinforcement learning algorithm for the first time. Then, it is used to implement a secure treatment decision-making algorithm. Simulation results show that our secure computation protocols and algorithms are feasible.

1. Introduction

As a recent healthcare tendency, personalized medicine [1] enables the patient to obtain early diagnoses, risk estimation, optimal treatments with low costs by using molecular and cellular analysis technologies, diagnosis results, genetic information, etc. Personalized medicine is usually implemented by the dynamic treatment regime technology [2, 3], which can provide various therapeutic methods according to the time-varying clinical states of the patient. This technology is particularly suitable for coping with complex chronic illnesses, such as diabetes, mental diseases, alcohol dependence, and human immunodeficiency virus infection, which have various stages.

Reinforcement learning [4], which is implemented by trial-and-error and interaction with the dynamic environment, is an important method for developing dynamic treatment regimes, industry automation, vehicular networks [5, 6], and other scenarios [7–12]. Meanwhile, with the developing technologies of internet of things and cloud computing, dynamic treatment regimes that are based on reinforcement learning are becoming increasingly attractive. For example, wearable devices are helpful for monitoring the patient's health data, which include heart rates and blood sugar levels. Next, collected health data are stored on the cloud. Then, the reinforcement learning algorithm can be implemented on these health data for making treatment decisions.

Unfortunately, because of the patient's limited computation ability, health data are usually outsourced to the cloud server for implementing the reinforcement learning algorithm. Because the cloud server may be untrusted, it is likely that health data will be illegally accessed, forged, tampered, or discarded in the process of transmission and computation. In addition, it may be harmful for personal privacy, economic interests, and even the security of human life. For example, as a billing service company, American Medical Collection Agency was intruded in 2019 [13]. This attack affects the health data of about 12 million patients. Besides, the parent firm of this company has filed for bankruptcy. Furthermore, unlike financial data or other types of human-generated data [14], health data are permanent biological data. They cannot be modified or wiped to avoid the damage, which is caused by health data disclosure.

In order to protect health data, we can encrypt health data by using a traditional encryption algorithm. Unfortunately, the reinforcement learning algorithm cannot be executed on the encrypted health data easily and flexibly. Homomorphic encryption [15] supports the operations on the ciphertext. Hence, the cloud server can run the reinforcement learning algorithm on the encrypted health data perfectly by using homomorphic encryption without leaking patient privacy. Finally, the encrypted computation result is returned to the patient. The computation result can be obtained by using the patient's secret key.

In this paper, we endeavor to study the security of health data in the above realistic scenario and focus on the secure implementation of the asynchronous advantage actor-critic (A3C) reinforcement learning algorithm. Taking into account the privacy and computation of health data on the untrusted cloud servers, we adopt homomorphic encryption as the main encryption primitive to carry out our research. Eventually, we make the following three contributions:

(1) Because the efficiency of Cheon et al.'s approximate homomorphic encryption scheme [16] is better than that of fully homomorphic encryption (FHE), we use it to design secure computation protocols, namely, homomorphic comparison protocol, homomorphic maximum protocol, homomorphic exponential protocol, and homomorphic division protocol. Based on these protocols, we first design the homomorphic reciprocal of square root protocol, which needs only one approximate computation

(2) Based on the proposed secure computation protocols, we design the secure A3C reinforcement learning algorithm for the first time. Then, we use it to implement a secure treatment decision-making algorithm

(3) Finally, we simulate the proposed secure computation protocols and algorithms on the personal computer's virtual machine. Then, we demonstrate the efficiency of our secure computation algorithms according to the thorough analysis

The layout of this paper is as follows. Section 2 analyzes related work about homomorphic encryption and secure computation of encrypted health data. Preliminaries are presented in Section 3. Section 4 shows related work about secure dynamic treatment regimes on health data. Building blocks are discussed in Section 5. Section 6 describes the proposed privacy-preserving A3C reinforcement learning algorithm and treatment decision-making algorithm. Performance results are shown and analyzed in Section 7. Finally, this paper is concluded in Section 8.

2. Related Work

In this section, we introduce related work about reinforcement learning, homomorphic encryption, and the computation of encrypted health data, which are described as follows.

Reinforcement learning can be mainly classified as value-based algorithms, policy-based algorithms, and actor-critic algorithms. Value-based algorithms usually compute the optimum cumulative reward and give a suggested policy. As a typical value-based algorithm, Q-learning is used for estimating the utility of the individual pair that consists of a state and an action. Q-learning has been applied for path planning [17, 18] in vehicular networks. Policy-based algorithms can evaluate the optimum policy directly. Williams [19] proposed a policy-based algorithm REINFORCE. Actor-critic algorithms combine the advantages of value-based algorithms and policy-based algorithms. The A3C reinforcement learning algorithm [20] is an actor-critic algorithm. It can work in discrete action spaces as well as continuous action spaces [21].

The concept of homomorphic encryption begins from privacy homomorphism [15]. According to the types of supported homomorphic operations, homomorphic encryption can be divided into partial homomorphic encryption (PHE), somewhat homomorphic encryption (SWHE), and FHE. PHE only supports homomorphic addition or homomorphic multiplication. SWHE is the basis of FHE. SWHE supports finite homomorphic addition and homomorphic multiplication. FHE supports arbitrary homomorphic addition and homomorphic multiplication.

In 2009, Gentry [22] designed the first FHE scheme, which is based on ideal lattices. Since then, homomorphic encryption has become a research hotspot. Next, in order to improve the efficiency of homomorphic operations, Gentry et al. [23] first constructed the FHE scheme, which is based on the approximate eigenvector method. In this scheme, the ciphertext noise increases linearly after each homomorphic multiplication. Although homomorphic multiplication of this scheme is efficient, it does not support the technique of single instruction multiple data (SIMD) [24]. Then, based on the learning with errors over rings (RLWE) [25] assumption and relinearization technique [24], Brakerski et al. [24] designed a FHE scheme, which supports the SIMD technique. However, this scheme does not support approximate homomorphic operations. Hence, based on Brakerski et al.'s scheme [24], Cheon et al. [16] proposed an improved homomorphic encryption scheme.

In terms of the computation of encrypted health data by using homomorphic encryption, there exist following several schemes. Khedr and Gulak [26] first proposed an optimized homomorphic encryption scheme, which is based on Gentry's scheme [23]. Then, the proposed scheme is applied for secure medical computations, which include comparison, Pearson goodness-of-fit test, and logistic regression. Sun et al. [27] implemented secure average heart rate, long QT syndrome detection, and chi-square tests by using Dowlin et al.'s FHE scheme [28]. Based on Boneh et al.'s homomorphic encryption scheme [29], Poon et al. [30] implemented the secure Fisher's exact test algorithm, which is often used to guarantee the statistical stability of genetic analysis. Raisaro et al. [31] used homomorphic encryption to explore genomic cohorts securely in a real scenario. In 2019, based on the distributed two trapdoors public-key algorithm [32] and Q-learning algorithm, Liu et al. [2] constructed a secure reinforcement learning model, which is helpful for making treatment decisions dynamically. Based on Fan's SWHE scheme [33], Jiang et al. [34] performed secure and efficient feature point detection and image matching for retinal images of diabetic retinopathy.

However, most of the above schemes are based on PHE, which only supports homomorphic addition or homomorphic multiplication. PHE may not support homomorphic multiplication. If homomorphic multiplication is required in some schemes, excessive rounds of interactions are needed. FHE can avoid this problem. But FHE confronts the problem of the efficiency of homomorphic operations. Furthermore, the Q-learning algorithm is usually used as the reinforcement learning algorithm in the above schemes. There does not exist an approach that can implement the A3C reinforcement learning algorithm securely.

3. Preliminaries

In this section, we begin with basic notations and definition of Cheon et al.'s approximate homomorphic encryption scheme. Then, we give the introduction of the A3C algorithm.

3.1. Basic Notations. Let $[z] = (z - 1/2, z + 1/2]$, where z is a real number. Let $[z]_p = z - [z/p] \cdot p \in (-p/2, p/2]$, where p denotes an integer.

Let $R = \mathbb{Z}/\langle \Phi_m(x) \rangle$ denote the ring modulo $\Phi_m(x)$, where λ is the security parameter, m is a positive integer, and $\Phi_m(x)$ is the mth cyclotomic polynomial. $R_q = \mathbb{Z}_q[x]/\langle \Phi_m(x) \rangle$ represents the ring modulo q and $\Phi_m(x)$, where q is the prime modulus, $q \geq 2$.

As for an integer $h > 0$, the distribution $\mathscr{HWT}(h)$ is selected from $\{0,\pm1\}$ randomly with the Hamming weight h. As for a rational number $\sigma > 0$, the distribution $\mathscr{DG}(\sigma^2)$ outputs a vector, which coefficients are selected from the discrete Gaussian distribution with the variance σ^2. As for a rational number $0 \leq \rho \leq 1$, the distribution $\mathscr{ZO}(\rho)$ is chosen from $\{0,\pm1\}$ randomly, where $\rho/2$ is the probability that ±1 is selected and $1 - \rho$ is the probability that 0 is selected.

3.2. Learning with Errors over Rings. In 2010, Lyubashevsky et al. [25] first proposed the RLWE assumption, which is described as follows.

Definition 1 (RLWE). The $\mathrm{RLWE}_{\lambda,q,\chi}$ assumption is to distinguish two distributions, namely, $(a, a \cdot s + e) \in R_q \times R_q$ and $(a, c) \in \mathrm{Unif}(R_q \times R_q)$, where $a \in R_q$ and $s \in R_q$, e is an error term, and Unif represents uniform random. Lyubashevsky et al. [25] proved that the security of RLWE assumption relies on ideal lattices.

3.3. Cheon et al.'s Homomorphic Encryption Scheme. In this subsection, we introduce Cheon et al.'s approximate homomorphic encryption scheme AHE = (KeyGen, Enc, Add, Sub, Mul, Dec, ReScale, Ecd, Dcd) [16] as follows:

(i) AHE.KeyGen$(1^\lambda, p, L)$: given the security parameter λ, an integer p, and a level L, this algorithm first sets $q_l = p^l \cdot q_0$, where q_0 is a fixed integer, $l = L, \cdots, 1$. It selects a power-of-two integer $M = M(\lambda, q_L)$, an integer $P = P(\lambda, q_L)$, and a rational number $\sigma = \sigma(\lambda, q_L)$. Next, it chooses a vector s from $\mathscr{HWT}(h)$. The secret key sk is set as $(1, s)$. A ring element a is sampled from R_{q_L}. An error term e is sampled from $\mathscr{DG}(\sigma^2)$. The public key pk is set as $(b, a) \in R_{q_L}^2$, where $b = -a \cdot s + e (\mathrm{mod}\ q_L)$. Then, a ring element a' is sampled from $R_{P \cdot q_L}$. An error term e' is sampled from $\mathscr{DG}(\sigma^2)$. The evaluation key evk is set as $(b', a') \in R_{P \cdot q_L}^2$, where $b' = -a' \cdot s + e' (\mathrm{mod}\ P \cdot q_L)$

(ii) AHE.Enc(pk, m): in order to encrypt a plaintext m, this algorithm samples an integer v from $\mathscr{ZO}(0.5)$. In addition, it chooses two error terms e_0 and e_1 from $\mathscr{DG}(\sigma^2)$. m is encrypted as the ciphertext $c = v \cdot \mathrm{pk} + (m + e_0, e_1) (\mathrm{mod}\ q_L)$

(iii) AHE.Dec(sk, c): in this algorithm, $c = (b, a)$ is decrypted as $b + a \cdot s (\mathrm{mod}\ q_l)$

(iv) AHE.Add(c', c''): in this algorithm, input parameters include two ciphertexts $c' = ([c'_0]_{q_l}, [c'_1]_{q_l})$ and $c'' = ([c''_0]_{q_l}, [c''_1]_{q_l})$, which are under the same secret key. Then, the additive ciphertext $c_{\mathrm{add}} = ([c_0]_{q_l} + [c''_0]_{q_l}, [c_1]_{q_l} + [c''_1]_{q_l})$

(v) AHE.Mul(evk, c', c''): in this algorithm, input parameters include evk, two ciphertexts $c' = ([c'_0]_{q_l}, [c'_1]_{q_l})$ and $c'' = ([c''_0]_{q_l}, [c''_1]_{q_l})$, where c' and c'' are under the same secret key. Then, the ciphertext $c_{\mathrm{temp}} = (c_0, c_1, c_2) = ([c'_0 \cdot c''_0]_{q_l}, [c'_0 \cdot c''_1 + c'_1 \cdot c''_0]_{q_l}, [c'_1 \cdot c''_1]_{q_l})$. The multiplicative ciphertext $c_{\mathrm{mul}} = (c_0, c_1) + [P^{-1} \cdot c_2 \cdot \mathrm{evk}] (\mathrm{mod}\ q_l)$

(vi) AHE.ReScale$_{l \longrightarrow l'}(c)$: as for a ciphertext $c \in R_{q_l}^2$ at the level l, the new ciphertext $c' = [(q_l'/q_l) c] (\mathrm{mod}\ q'_l)$

(vii) AHE.Ecd(z, Δ): as for a vector $z = (z_0, z_1, \cdots, z_{N/2-1}) \in \mathbb{C}^{N/2}$ and a scaled factor $\Delta > 0$, this algorithm outputs $z' = [\sigma^{-1}(\Delta \cdot z)] \in R$, where σ^{-1} is the inverse operation of a canonical embedding map $\sigma(\cdot)$

(viii) AHE.Dcd(z', Δ): as for $z' \in R$, this algorithm outputs $z = \Delta^{-1} \cdot \sigma(m) \in \mathbb{C}^{N/2}$

In Cheon et al.'s scheme, the decryption noise should be bounded by $8\sqrt{2} \cdot \sigma \cdot N + 6\sigma \cdot \sqrt{N} + 16\sigma \cdot \sqrt{h \cdot N}$ for the correctness of decryption. In addition, the noise of the rescaling ciphertext is at most $\sqrt{N/3} \cdot (3 + 8\sqrt{h})$. Furthermore, the noise of the multiplicative ciphertext should be less than $P^{-1} \cdot q_l \cdot 8\sigma \cdot N/\sqrt{3} + \sqrt{N/3} \cdot (3 + 8\sqrt{h})$. The details about the analysis of Cheon et al.'s scheme can be found in [16].

3.4. Asynchronous Advantage Actor-Critic Reinforcement Learning Algorithm. In 2016, Mnih et al. [20] proposed the asynchronous advantage actor-critic reinforcement learning algorithm, which is based on combining the value-based method and the policy-based method. One advantage of the A3C algorithm is that it can work in discrete action spaces as well as continuous action spaces. In addition, in order to improve the learning efficiency of the A3C algorithm, multiple asynchronous actor-learners, which can interact with the environment and acquire various independent exploration policies, are running in parallel. The details of the A3C algorithm are described as follows.

In the A3C algorithm, there is a policy function $\pi(a_t \mid s_t ; \theta)$ and a value function $V(s_t ; \theta_v)$, where a_t denotes an action at the time step t, s_t denotes a state at the time step t, and θ and θ_v are two parameters. In addition, $V(s_t ; \theta_v)$ and $\pi(a_t \mid s_t ; \theta)$ will be updated $t_{\max}$ times, where $t_{\max}$ denotes the maximum step. $V(s_t ; \theta_v)$ and $\pi(a_t \mid s_t ; \theta)$ are usually approximated by a single convolutional neural network. Specifically, $V(s_t ; \theta_v)$ is based on a linear layer. $\pi(a_t \mid s_t ; \theta)$ is relied on a softmax layer. Namely, $V(s_t ; \theta_v) = x(s_t) \cdot \theta_v$, where $x(s_t)$ is a function which is related to s_t. $\pi(a_t \mid s_t ; \theta) = e^{f(a_t|s_t)\cdot\theta} / \sum_{j=0}^{t_{\max}} e^{f(a_j|s_j)\cdot\theta}$, a_j is an action at the time step j, and $f(a_j \mid s_j)$ is a function which is related to a_j and s_j.

Furthermore, the A3C algorithm uses two loss functions, namely, policy loss function and value loss function, which are described as follows. On the one hand, the policy loss function

$$f_\pi(\theta) = \ln \pi(a_t \mid s_t ; \theta) \cdot (R - V(s_t ; \theta_v)) + \beta \cdot H(\pi(s_t ; \theta)), \tag{1}$$

where R is the reward and the parameter k depends on the state. In addition, the upper bound of k is $t_{\max}$. r_{t+i} is the immediate reward. The discount factor $\gamma \in (0, 1]$. The entropy function $H(\pi(s_t ; \theta))$ can be set as $-\Sigma_{i=0}^{k} f(a_t \mid s_t) \cdot \theta \cdot \ln \pi(s_t ; \theta)$. The hyperparameter β can adjust the intensity of the entropy regulation term. Then, we can conclude that

$$\begin{aligned} f_\pi(\theta) = {} & \left(f(a_t \mid s_t) - \Sigma_{j=0}^{t_{\max}} f\left(a_j \mid s_j\right)\right) \cdot \theta \cdot (R - x(s_t) \cdot \theta_v) \\ & + \beta \cdot \left(-\Sigma_{i=0}^{k} f(a_t \mid s_t) \cdot \left(f(a_t \mid s_t) - \Sigma_{j=0}^{t_{\max}} f\left(a_j \mid s_j\right)\right) \cdot \theta^2\right). \end{aligned} \tag{2}$$

Hence, the differentiation of $f_\pi(\theta)$ with respect to θ is

$$\begin{aligned} \frac{\partial f_\pi(\theta)}{\partial \theta} = {} & \left(f(a_t \mid s_t) - \Sigma_{j=0}^{t_{\max}} f\left(a_j \mid s_j\right)\right) \\ & \cdot (R - x(s_t) \cdot \theta_v) + 2\beta \cdot \theta \cdot f(a_t \mid s_t) \\ & \cdot \left(f(a_t \mid s_t) - \Sigma_{j=0}^{t_{\max}} f\left(a_j \mid s_j\right)\right). \end{aligned} \tag{3}$$

On the other hand, the value loss function

$$f_v(\theta_v) = (R - V(s_t ; \theta_v))^2 = (R - x(s_t) \cdot \theta_v)^2. \tag{4}$$

Hence, the differentiation of $f_v(\theta_v)$ with respect to θ_v is

$$\frac{\partial f_v(\theta_v)}{\partial \theta_v} = 2(R - x(s_t) \cdot \theta_v)\left(\frac{\partial R}{\partial \theta_v} - x(s_t)\right). \tag{5}$$

Based on the above two loss functions and corresponding differentiation, the A3C reinforcement learning algorithm is defined in Algorithm 1, which is described as follows. Algorithm 1 requires input parameters θ, θ_v, θ', θ'_v, T, t, $t_{\max}$, $T_{\max}$, t_g, η, W, and α, where the definition of these parameters are shown in Table 1. In order to implement Algorithm 1, we first set $T = 0$, $t = 1$. If $T < T_{\max}$ and $w \in [1, W]$, we implement the iteration, which is shown as follows. Global gradients $d\theta$ and $d\theta_v$ are set as 0. θ' and θ'_v are synchronized as θ and θ_v, respectively. We set $t_0 = t$ and obtain the system state $\mathcal{S}_t \in \mathcal{S}$, where $\mathcal{S}$ is a state set, $\mathcal{S} = (\mathcal{S}_0, \cdots, \mathcal{S}_{\varphi-1})$, and φ is the number of states. Next, we repeat a subalgorithm until $t - t_0 \neq t_{\max}$. In this subalgorithm, the action $\mathcal{A}_t \in \mathcal{A}$ is obtained by using $\pi(\mathcal{A}_t \mid \mathcal{S}_t ; \theta')$, where $\mathcal{A}$ is an action set, $\mathcal{A} = (\mathcal{A}_0, \cdots, \mathcal{A}_{\chi-1})$, and χ is the number of actions. We execute $\mathcal{A}_t$, get the reward R_t, and observe the next state $\mathcal{S}_{t+1}$, where R_t is set as $\Sigma_{i=0}^{k-1} \gamma^i \cdot r_{t+i} + \gamma^k \cdot V(\mathcal{S}_{t+k} ; \theta'_v) = \Sigma_{i=0}^{k-1} \gamma^i \cdot r_{t+i} + \gamma^k \cdot x(\mathcal{S}_{t+k}) \cdot \theta'_v$. In addition, we set $t = t + 1$. After the implementation of the above subalgorithm, we observe whether $t\% t_g$ equals to 0. If $\mathcal{S}_t$ is terminal, we set $R = 0$. If $\mathcal{S}_t$ is nonterminal $R = V(\mathcal{S}_t ; \theta'_v) = x(\mathcal{S}_t) \cdot \theta'_v$. Then, we repeat a subalgorithm from $i = t - 1$ to $i = t_0$. In this subalgorithm, R is set as $R_t + \gamma \cdot R$, namely, $R = \Sigma_{i=0}^{k-1} \gamma^i \cdot r_{t+i} + \gamma^k \cdot x(\mathcal{S}_t) \cdot \theta'_v + \gamma \cdot R$. We compute

$$\begin{aligned} \frac{\partial f_\pi\left(\theta'\right)}{\partial \theta'} = {} & \left(f(\mathcal{A}_t \mid \mathcal{S}_t) - \sum_{j=0}^{t_{\max}} f(\mathcal{A}_t \mid \mathcal{S}_t)\right) \\ & \cdot (R - x(\mathcal{S}_t) \cdot \theta_v) + 2\beta \cdot \theta' \cdot f(\mathcal{A}_t \mid \mathcal{S}_t) \\ & \cdot \left(f(\mathcal{A}_t \mid \mathcal{S}_t) - \sum_{j=0}^{t_{\max}} f(\mathcal{A}_t \mid \mathcal{S}_t)\right), \end{aligned} \tag{6}$$

$A3C(\theta, \theta_v, \theta', \theta'_v, T, t, t_{\max}, T_{\max}, t_g, \eta, W, \alpha)$:
Input: $\theta, \theta_v, \theta', \theta'_v, T, t, t_{\max}, T_{\max}, t_g, \eta, W, \alpha$.
Output: θ, θ_v.
Set $T = 0$, $t = 1$.
While $T < T_{\max}$ do.
  For $w = 1$ to W do.
    Set $d\theta = 0$, $d\theta_v = 0$.
    Synchronize $\theta' = \theta$, $\theta'_v = \theta_v$.
    Set $t_0 = t$ and get $\mathcal{S}_t$.
    Repeat.
      Get $\mathcal{A}_t$ according to $\pi(\mathcal{A}_t \mid \mathcal{S}_t; \theta')$.
      Execute $\mathcal{A}_t$, get R_t and observe $\mathcal{S}_{t+1}$.
      $t = t + 1$.
    Until $t - t_0 = t_{\max}$
    If $\mathcal{S}_t$ is terminal, $R = 0$.
    If $\mathcal{S}_t$ is non-terminal, $R = x(\mathcal{S}_t) \cdot \theta'_v$.
    For $i = t - 1$ to t_0 do
      $R = R_t + \gamma \cdot R$.
      Compute $\partial f_\pi(\theta')/\partial\theta' = (f(\mathcal{A}_t \mid \mathcal{S}_t) - \sum_{j=0}^{t_{\max}} f(\mathcal{A}_t \mid \mathcal{S}_t))$ $(R - x(\mathcal{S}_t) \cdot \theta_v) + 2\beta \cdot \theta' f(\mathcal{A}_t \mid \mathcal{S}_t) \cdot (f(\mathcal{A}_t \mid \mathcal{S}_t) - \sum_{j=0}^{t_{\max}}$ $f(\mathcal{A}_t \mid \mathcal{S}_t))$.
      Compute $d\theta = d\theta + (\partial f_\pi(\theta')/\partial\theta')$.
      Compute $\partial f_v(\theta'_v)/\partial\theta'_v = 2(R - x(\mathcal{S}_t) \cdot \theta'_v) \cdot (\partial R/\partial\theta'_v)$.
      Compute $d\theta_v = d\theta_v + (\partial f_\pi(\theta'_v)/\partial\theta'_v)$.
    End for.
    Compute $g = \alpha \cdot g + (1 - \alpha)(d\theta)^2$, $g_v = \alpha \cdot g_v + (1 - \alpha)(d\theta_v)^2$.
    Compute $\theta = \theta - \eta(d\theta/\sqrt{g + \varepsilon})$, $\theta_v = \theta_v - \eta(d\theta_v/\sqrt{g_v + \varepsilon})$.
  End for
End while

Algorithm 1: A3C reinforcement learning algorithm.

Table 1: Notations.

Symbol	Description
θ, θ_v	Shared parameter vectors in the global network
θ', θ'_v	Thread-specific parameter vectors in the local network
T	Global counter
t	Local step counter
$t_{\max}, T_{\max}$	Upper bounds
t_g	An integer
η	Learning rate
W	Number of agents
α	Momentum

where $\partial f_\pi(\theta')/\partial\theta'$ is the differentiation of $f_\pi(\theta')$ with respect to θ'. $d\theta$ is set as $d\theta + \partial f_\pi(\theta')/\partial\theta'$. We compute

$$\frac{\partial f_v\left(\theta'_v\right)}{\partial\theta'_v} = 2\left(R - x(\mathcal{S}_t) \cdot \theta'_v\right) \cdot \frac{\partial R}{\partial\theta'_v}, \tag{7}$$

where $\partial f_v(\theta'_v)/\partial\theta'_v$ is the differentiation of $f_v(\theta'_v)$ with respect to θ'_v and $\partial R/\partial\theta'_v$ is the differentiation of R with respect to θ'_v. Finally, θ and θ_v can be updated by using equations $\theta = \theta - \eta(d\theta/\sqrt{g + \varepsilon})$ and $\theta_v = \theta_v - \eta(d\theta_v/\sqrt{g_v + \varepsilon})$, respectively, where $g = \alpha \cdot g + (1 - \alpha)(d\theta)^2$ and $g_v = \alpha \cdot g_v + (1 - \alpha)(d\theta_v)^2$.

4. Secure Dynamic Treatment Regimes on Health Data

4.1. System Model. As shown in Figure 1, the system model of secure dynamic treatment regimes on health data consists of four parts, namely, undiagnosed patient, key generation center, cloud servers, and historical data owners, which are described as follows:

(i) The undiagnosed patient's current state is collected by using wearable devices, which integrate modules of physiological sensors, weak computation, and communication. Wearable devices include smart bracelet, smart glasses, sleep monitoring sensors, and smart watch. They can collect a variety of health data, such as body temperature, heart rate, blood sugar, and blood volume index. Then, these health

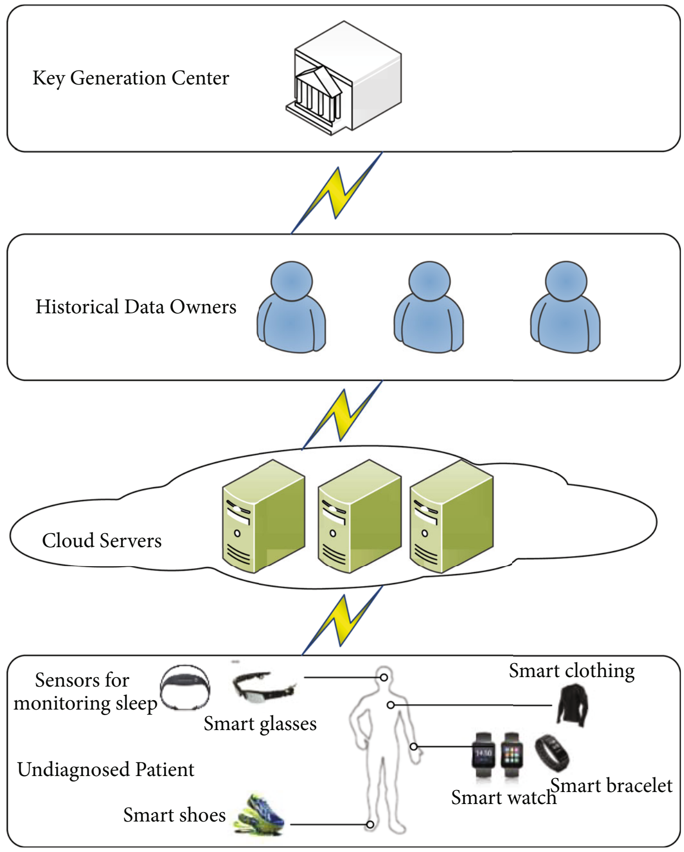

Figure 1: The model of secure dynamic treatment regimes on health data.

data are transmitted to cloud servers for computation. Based on the returned computation result, the patient can obtain the diagnosis

(ii) Key generation center is an indispensable and independent entity, which is trusted by the other entities in this system model. It is responsible for distributing and managing all the public keys and private keys of Cheon et al.'s homomorphic encryption scheme for wearable devices and undiagnosed patients via a secure channel

(iii) Cloud servers have the powerful data storage space. Hence, they store and manage ciphertexts, which come from undiagnosed patients and wearable devices. Additionally, they can perform some computations on these ciphertexts

(iv) Historical data owners have a sequence of medical data and their corresponding decision results. These encrypted data are transmitted and stored on cloud servers. Cloud servers can compute these ciphertexts for training the reinforcement learning model

4.2. Attack Model. In this paper, we suppose that the entities in the system model are honest-but-curious. Namely, the entities strictly follow the designed protocols. But they are interested in acquiring medical data of other entities. We suppose that there is an adversary $\mathbf{A}_1^*$ in the attack model. The goal of $\mathbf{A}_1^*$ is to guess the plaintexts of the challenge historical data owners' ciphertexts or the challenge wearable devices' ciphertexts.

In order to acquire the ciphertexts of historical data owners and wearable devices, middle ciphertext results during the execution of privacy-preserving A3C reinforcement learning algorithm and treatment decision-making algorithm (Section 6), $\mathbf{A}_1^*$ eavesdrops on the communication links among the entities in the system model. However, these ciphertexts are based on Cheon et al.'s approximate homomorphic encryption scheme [16]. Hence, $\mathbf{A}_1^*$ cannot decrypt these ciphertexts without knowing their secret keys. It can be guaranteed by using the semantic security of Cheon et al.'s scheme. In addition, the key generation center distributes key pairs to historical data owners and wearable devices in a secure way. Furthermore, due to the lack of private keys of these ciphertexts, $\mathbf{A}_1^*$ cannot generate evaluation keys. Hence, $\mathbf{A}_1^*$ cannot transform these ciphertexts into some domains that $\mathbf{A}_1^*$ can decrypt. Besides, $\mathbf{A}_1^*$ cannot get useful information by adding or multiplying a plaintext with these ciphertexts. In a conclusion, the proposed model is secure.

4.3. System Setup and Overview. Our secure model of dynamic treatment regimes consists of two phases, which are described as follows.

(i) Training dataset outsourcing and initialization: historical data owners initialize input parameters θ, θ_v, learning rate η, and discount factor γ. The state set $\mathcal{S} = (\mathcal{S}_0,\cdots,\mathcal{S}_{\varphi-1})$ and action set $\mathcal{A} = (\mathcal{A}_0,\cdots,\mathcal{A}_{\chi-1})$ are encrypted as $c_{\mathcal{S}} = (c_{\mathcal{S}_0},\cdots,c_{\mathcal{S}_{\varphi-1}})$ and $c_{\mathcal{A}} = (c_{\mathcal{A}_0},\cdots,c_{\mathcal{A}_{\chi-1}})$. Then, historical data owners send $c_{\mathcal{S}}$, $c_{\mathcal{A}}$, η, γ, and other parameters to cloud servers for storage and computation, where $j = 0, \cdots, n-1$, and n is the number of historical data owners

(ii) Outsourced sequential treatment decision making: in order to achieve sequential treatment decision making, the undiagnosed patient's current state x, which comes from wearable devices, is encrypted as c_x. Then, c_x is transmitted to cloud servers for treatment decision making. Based on our privacy-preserving A3C reinforcement learning algorithm and treatment decision-making algorithm, cloud servers output the encrypted treatment decision c_a. The undiagnosed patient decrypts c_a to obtain the treatment decision a by using his own secret key

5. Building Blocks

5.1. Encoding Rational Number. In order to implement the privacy-preserving A3C reinforcement learning algorithm, we need to encrypt health data. Health data are usually rational numbers. However, most of the homomorphic encryption schemes only support homomorphic operations over integers. They cannot cope with rational numbers. Hence, in this paper, we take Cheon et al.'s encoding technique [16], which can encode a rational number. Then, the rational number can be converted to a ring element just like using the integer encoding technique. We can use Cheon et al.'s scheme [16] to encrypt the converted result.

5.2. Homomorphic Comparison Protocol. In order to implement comparison for our secure computation algorithms, we design a homomorphic comparison protocol by using Cheon et al.'s scheme [16] and Sun et al.'s method [35]. We suppose that the user owns plaintexts m_0 and m_1. Then, the user uses Cheon et al.'s scheme to encrypt these plaintexts. The ciphertexts are c_0 and c_1, respectively. The user owns the secret key sk. The cloud server is responsible for storing the ciphertexts. As shown in Algorithm 2, the cloud server first computes the ciphertext $c_b = t + c_0 - c_1$, where t is the plaintext modulus. Next, the cloud server transmits c_b to the user. The user uses sk to decrypt c_b. The decryption result is b. If $b > t$, $m_0 > m_1$. If $b = t$, $m_0 = m_1$. If $b < t$, $m_0 < m_1$. For example, we suppose that $t = 2$, $m_0 = 0$, $m_1 = 1$, and then $c_b = 2 + c_0 - c_1$, where c_0 and c_1 are ciphertexts of 0 and 1, respectively. The decryption result of c_b equals to 1. Hence, $m_0 < m_1$.

5.3. Homomorphic Maximum Protocol. In order to compute the encrypted index of the largest plaintext, we design a homomorphic maximum protocol by using the above protocol and Sun et al.'s method [35]. We suppose that the user owns m_0, $\cdots$, m_{k-1}, where k is the number of plaintexts. The user uses Cheon et al.'s scheme [16] to encrypt these plaintexts. The ciphertexts are c_0, $\cdots$, c_{k-1}, respectively. The user owns the secret key sk. As shown in Algorithm 3, the cloud server computes $c_b = \mathrm{comp}(c_{\max}, c_i)$, where $c_{\max}$ is initialized as c_0. If the decryption result $b < t$, $c_{\max} = c_i$, $i = i + 1$. The cloud server continues to compare $c_{\max}$ and c_i until $i > k - 1$. Finally, the user can obtain the index of the largest plaintext by decrypting $c_{\max}$. For example, there exist ciphertexts c_2, c_3, and c_4, whose plaintexts are 2, 3, and 4, respectively. We set $c_{\max} = c_2$. Next, we first compare $c_{\max}$ and c_3. $c_{\max}$ is updated as c_3. Then, we compare $c_{\max}$ and c_4. $c_{\max}$ is updated as c_4. After the decryption of $c_{\max}$, the user gets the maximum result 4.

comp(c_0, c_1):
Input: c_0 and c_1.
Output: c_b.
1. Compute $c_b = t + c_0 - c_1$.
2. Return c_b to the user.
3. If $b > t$, $m_0 > m_1$.
   If $b = t$, $m_0 = m_1$.
   If $b < t$, $m_0 < m_1$.

Algorithm 2: Homomorphic comparison protocol.

argmax($c_0, c_1, \cdots, c_{k-1}$):
Input: $c_0, c_1, \cdots, c_{k-1}$.
Output: $c_{\max}$.
1. Set $c_{\max} = c_0$.
2. For $i = 1$ to $k - 1$ do.
3. $c_b = \mathrm{comp}(c_{\max}, c_i)$.
4. Decrypt c_b to get b.
5. If $b = 1$, $c_{\max} = c_i$.
6. $i = i + 1$.
End for

Algorithm 3: Homomorphic maximum protocol.

5.4. Homomorphic Exponential Protocol. In this section, based on the Taylor series, we begin to describe the homomorphic exponential protocol. We suppose that the user owns the plaintext m. Next, it is encrypted as c_m by using Cheon et al.'s homomorphic encryption scheme. Only the user has the secret key. Then, c_m is stored on the cloud server. In the homomorphic exponential protocol (Algorithm 4), the cloud server first computes the ciphertext $c_{e^m} = 1 + c_m + c_m^2/2! + c_m^3/3! + \cdots + c_m^n/n!$ without decryption, where n denotes an integer. The precision of e^m increases with the increasing of n. Then, c_{e^m} is returned to the user. The user gets the exponential result e^m by using his secret key. For example, we can set $m = 4$, $n = 3$, and then $c_{e^2} = 1 + c_4 + c_4^2/2! + c_4^3/3!$, where c_{e^2} and c_4 are ciphertexts of e^2 and 4, respectively. After the decryption of c_{e^2}, the user gets the exponential result e^2.

5.5. Homomorphic Division Protocol. In this section, we begin to describe the homomorphic division protocol. We suppose that the user owns plaintexts m_0, m_1, and m_2. Then, they are encrypted as c_{m_0}, c_{m_1}, and c_{m_2} by using Cheon et al.'s

exp (c_m, n):
Input: c_m, n.
Output: c_{e^m}.
1. Compute $c_{e^m} = 1 + c_m + \cdots + (c_m^n/n!)$.
2. Return c_{e^m} to the user.

Algorithm 4: Homomorphic exponential protocol.

div $(c_{m_0}, c_{m_1}, c_{m_2})$:
Input: c_{m_0}, c_{m_1} and c_{m_2}.
Output: The ciphertext c_{div}.
1. Compute $c_{\text{add}} = c_{m_0} + c_{m_1}$.
2. Return c_{add} to the user.
3. Decrypt c_{add} to obtain add.
4. Calculate rev = 1/add.
5. rev is encrypted as c_{rev}.
6. c_{rev} is transmitted to the cloud server.
7. Calculate $c_{\text{div}} = c_{m_2} \cdot c_{\text{rev}}$.
8. c_{div} is returned to the user.

Algorithm 5: Homomorphic division protocol.

homomorphic encryption scheme. Only the user has the secret key. Then, c_{m_0}, c_{m_1}, and c_{m_2} are transmitted to the cloud server. In order to output the ciphertext c_{div} of the plaintext $m_2/(m_0 + m_1)$, we design the homomorphic division protocol (Algorithm 5), which is described as follows. The cloud server first computes the ciphertext $c_{\text{add}} = c_{m_0} + c_{m_1}$ without decryption, where the plaintext of c_{add} is add $= m_0 + m_1$. Then, c_{add} is returned to the user. The user gets the plaintext add by using his secret key. The user calculates rev = 1/add. rev is encrypted as c_{rev} by using Cheon et al.'s scheme. c_{rev} is transmitted to the cloud server. The cloud server calculates the ciphertext $c_{\text{div}} = c_{m_2} \cdot c_{\text{rev}}$. Finally, c_{div} is returned to the user. After the decryption of c_{div}, the user gets the division result div $= m_2/(m_0 + m_1)$. For example, we set $m_0 = 6$, $m_1 = 1$, and $m_2 = 2$. Then, the cloud server calculates $c_{\text{add}} = c_{m_0} + c_{m_1} = c_3$, where the plaintext of c_3 is 3. c_3 is returned to the user. After the decryption of c_3, the user calculates rev = 1/3 = 0.33. The ciphertext c_{rev} is sent to the cloud server. The cloud server calculates $c_{\text{div}} = c_{m_2} \cdot c_{\text{rev}} = c_{0.33\times 6} = c_{1.98}$, where the plaintext of $c_{1.98}$ is 1.98.

5.6. Homomorphic Reciprocal of Square Root Protocol. In this section, we begin to describe the homomorphic reciprocal of square root protocol. The traditional method is to compute the ciphertext of the approximate square root firstly. Then, it computes the approximate reciprocal of square root homomorphically. However, two approximate computations will affect the precision of the final result. Hence, based on Lomont's fast inverse square root algorithm [36], we design a new homomorphic reciprocal of square root protocol (Algorithm 6), which only needs one approximate computation. In our protocol, we suppose that the user owns the floating number $m = (1 + m_0)2^{m_1 - 127}$, where $0 < m_0 < 1$ and

$1/\sqrt{c_m}$:
Input: The ciphertext c_m.
Output: $c_{1/\sqrt{m}}$.
1. Transmit c_m to the user.
2. Decrypt c_m to obtain $m = (1 + m_0)2^{m_1 - 127}$.
3. Convert m to obtain $m' = m_1 \cdot 2^{23} + m_0 \cdot 2^{23}$.
4. Encrypt m' as $c_m{}'$.
5. Transmit $c_m{}'$ to the cloud server.
6. Compute $c_{\text{temp}} = 3/2(127 - 0.045)2^{23} - 0.5c_m{}'$.
7. Transmit c_{temp} to the user.
8. Decrypt c_{temp} to obtain temp $= (\text{temp}_1 + \text{temp}_0) \cdot 2^{23}$.
9. Convert temp to temp$' = (1 + \text{temp}_0)2^{\text{temp}_1 - 127}$.
10. Encrypt temp$'$ as $c_{\text{temp}}{}'$.
11. Transmit $c_{\text{temp}}{}'$ to the cloud server.
12. Compute $c_{1/\sqrt{m}} = 3/2c_{\text{temp}}{}' - 1/2c_m \cdot c^3_{\text{temp}'}$.
13. Return $c_{1/\sqrt{m}}$ to the user.

Algorithm 6: Homomorphic reciprocal of square root protocol.

$0 < m_1 < 255$. It is encrypted as c_m by using Cheon et al.'s homomorphic encryption scheme. Only the user has the secret key. c_m is stored on the cloud server. c_m is first transmitted to the user. The user decrypts c_m by his secret key. The decryption result m is converted to an integer $m' = m_1 \cdot 2^{23} + m_0 \cdot 2^{23}$. m' is encrypted as $c_m{}'$ by using Cheon et al.'s scheme. Next, $c_m{}'$ is transmitted to the cloud server. The cloud server computes the intermediate ciphertext

$$c_{\text{temp}} = \frac{3}{2}(127 - 0.045)2^{23} - 0.5c_{m'}, \tag{8}$$

where the plaintext of c_{temp} is the floating number temp. Then, the cloud server sends c_{temp} to the user. The user decrypts c_{temp} to obtain temp $= \text{temp}_1 \cdot 2^{23} + \text{temp}_0 \cdot 2^{23}$, where $0 < \text{temp}_0 < 1$ and $0 < \text{temp}_1 < 255$. temp is converted to an integer temp$' = (1 + \text{temp}_0)2^{\text{temp}_1 - 127}$. temp$'$ is encrypted as $c_{\text{temp}}{}'$. $c_{\text{temp}}{}'$ is transmitted to the cloud server. The cloud server computes the ciphertext

$$c_{1/\sqrt{m}} = \frac{3}{2}c_{\text{temp}}{}' - \frac{1}{2}c_m \cdot c^3_{\text{temp}'}. \tag{9}$$

Then, $c_{1/\sqrt{m}}$ is returned to the user. The user gets the reciprocal of square root $1/\sqrt{m}$ by using his secret key. For example, we set $m = (1 + 0.25)^{124-127} = 0.156$. Then, m is converted to $m' = 62 \cdot 2^{23} + 0.125 \cdot 2^{23}$. The ciphertext $c_m{}'$ of m' is sent to the cloud server. The cloud server computes

$$c_{\text{temp}} = \frac{3}{2}(127 - 0.045)2^{23} - 0.5c_{m'}. \tag{10}$$

The user decrypts c_{temp} to obtain temp $= 128 \cdot 2^{23} + 0.3075 \cdot 2^{23}$. temp is converted to temp$' = (1 + 0.3075)$ $2^{128-127} = 2.615$. The ciphertext $c_{\text{temp}}{}'$ of temp$'$ is sent to

```
PA3C(θ, θ_v, θ', θ'_v, T, t, t_max, T_max, t_g, η, W, α, c_S, c_A):
Input: θ, θ_v, θ', θ'_v, T, t, t_max, T_max, t_g, η, W, α, c_S, c_A.
Output: θ, θ_v.
Set T = 0, t = 1.
While T < T_max do
  For w = 1 to W do
    Set dθ = 0, dθ_v = 0. Synchronize θ' = θ, θ'_v = θ_v.
    Set t_0 = t, get c_{S_t}.
    Repeat
      Get c_{A_t} according to argmax(c_{π_0},⋯,c_{π_{t_max}}).
      Execute c_{A_t}, get c_{R_t}.
      Observe c_{S_{t+1}}, t = t + 1.
    Until t − t_0 = t_max
    If c_{S_t} is terminal, c_R = 0.
    If c_{S_t} is non-terminal, c_R = x(c_{S_t}) · θ'_v.
    For i = t − 1 to t_0 do
      If c_{S_t} is terminal
        c_R = Σ_{i=0}^{k−1} γ^i · r_{t+i} + γ^k · x(c_{S_t}) · θ'_v.
        Compute c_{∂f_π(θ')/∂θ'}, c_{∂f_v(θ'_v)/∂θ'_v}.
      End if
      If c_{S_t} is non-terminal
        c_R = Σ_{i=0}^{k−1} γ^i · r_{t+i} + γ^k · x(c_{S_t}) · θ'_v + γ · x(c_{S_t}) · θ'_v.
        Compute c_{∂f_π(θ')/∂θ'}, c_{∂f_v(θ'_v)/∂θ'_v}.
      End if
      Compute c_{dθ} = c_{dθ} + c_{∂f_π(θ')/∂θ'}.
      Compute c_{dθ_v} = c_{dθ_v} + c_{∂f_π(θ'_v)/∂θ'_v}.
    End for
    c'_{dθ} and c'_{dθ_v} are returned to the cloud server.
    Compute c_g = αc_g + (1 − α)(c_{dθ})^2, c_{g_v} = αc_{g_v} + (1 − α)(c_{dθ_v})^2.
    c'_g and c'_{g_v} are sent to the cloud server.
    Set c_g = c'_g, c_{g_v} = c'_{g_v}.
    Compute c_θ = θ − η(c'_{dθ}/√(c_g + ε)), c_{θ_v} = θ_v − η(c'_{dθ_v}/√(c_{g_v} + ε)).
    c_θ and c_{θ_v} are returned to the user.
    Decrypt c_θ and c_{θ_v} to obtain θ and θ_v.
  End for
End while
```

Algorithm 7: Privacy-preserving A3C reinforcement learning algorithm.

the cloud server. The cloud server computes the ciphertext

$$c_{1/\sqrt{0.156}} = \frac{3}{2} c_{\mathrm{temp}'} - \frac{1}{2} c_m \cdot c^3_{\mathrm{temp}'}. \tag{11}$$

The user decrypts $c_{1/\sqrt{m}}$ to obtain $1/\sqrt{0.156} = 3/2 \cdot 2.615 - 1/2 \cdot 0.156 \cdot 2.615^3 \approx 2.528$.

6. Privacy-Preserving Computation Algorithms

6.1. Privacy-Preserving A3C Reinforcement Learning Algorithm. In this section, we begin to describe how to implement a privacy-preserving A3C reinforcement learning algorithm by using Cheon et al.'s approximate homomorphic encryption scheme [16]. As shown in Algorithm 7, we describe the privacy-preserving A3C reinforcement learning algorithm as follows. Algorithm 7 requires input parameters θ, θ_v, θ', θ'_v, T, t, $t_{\max}$, $T_{\max}$, t_g, η, W, α, $c_{\mathcal{S}}$, and $c_{\mathcal{A}}$. Set $T = 0$, $t = 1$. If $T < T_{\max}$ and $w \in [1, W]$, we implement the iteration, which is shown as follows. $d\theta$ and $d\theta_v$ are set as 0. θ' and θ'_v are synchronized as θ and θ_v, respectively. We set $t_0 = t$ and obtain the encrypted system state $c_{\mathcal{S}_t} \in c_{\mathcal{S}}$. Next, we repeat a subalgorithm until $t - t_0 \neq t_{\max}$. In this subalgorithm, the encrypted action $c_{\mathcal{A}_t} \in c_{\mathcal{A}}$ is obtained based on argmax(c_{π_0}, $c_{\pi_1},\cdots,c_{\pi_{t_{\max}}}$), where $c_{\pi_j} = e^{f(c_{\mathcal{A}_j}|c_{\mathcal{S}_j})\cdot\theta'}$, $j = 0, 1, \cdots, t_{\max}$. We execute $c_{\mathcal{A}_t}$ and get the encrypted reward

$$c_{R_t} = \Sigma_{i=0}^{k-1} \gamma^i \cdot r_{t+i} + \gamma^k \cdot x\left(c_{\mathcal{S}_{t+k}}\right) \cdot \theta_v. \tag{12}$$

We observe the next encrypted state $c_{\mathcal{S}_{t+1}} \in c_{\mathcal{S}}$. In addition, we set $t = t + 1$. After the implementation of the above subalgorithm, we observe whether $c_{\mathcal{S}_t}$ is terminal. Then, we repeat a subalgorithm from $i = t - 1$ to $i = t_0$. In this subalgorithm, if $c_{\mathcal{S}_t}$ is terminal, c_R is set as

$$c_R = \Sigma_{i=0}^{k-1} \gamma^i \cdot r_{t+i} + \gamma^k \cdot c_{x(\mathcal{S}_t)} \cdot \theta'_v. \tag{13}$$

The cloud server computes

$$\begin{aligned} c_{\frac{\partial f_\pi(\theta')}{\partial \theta'}} = {} & \left(f\left(c_{\mathcal{A}_t} \mid c_{\mathcal{S}_t}\right) - \Sigma_{j=0}^{t_{\max}} f\left(c_{\mathcal{A}_t} \mid c_{\mathcal{S}_t}\right) \right) \\ & \cdot \left(\Sigma_{i=0}^{k-1} \gamma^i \cdot r_{t+i} + \gamma^k \cdot x\left(c_{\mathcal{S}_t}\right) \cdot \theta'_v - x\left(c_{\mathcal{S}_t}\right) \cdot c_{\theta_v} \right) \\ & + 2\beta \cdot c_\theta{}' \cdot f\left(c_{\mathcal{A}_t} \mid c_{\mathcal{S}_t}\right) \\ & \cdot \left(f\left(c_{\mathcal{A}_t} \mid c_{\mathcal{S}_t}\right) - \Sigma_{j=0}^{t_{\max}} f\left(c_{\mathcal{A}_j} \mid c_{\mathcal{S}_j}\right) \right). \end{aligned} \tag{14}$$

The cloud server computes

$$\begin{aligned} c_{\frac{\partial f_v(\theta'_v)}{\partial \theta'_v}} = {} & 2\left(\Sigma_{i=0}^{k-1} \gamma^i \cdot r_{t+i} + \gamma^k \cdot x\left(c_{\mathcal{S}_t}\right) \cdot c_{\theta'_v} - x\left(c_{\mathcal{S}_t}\right) c_{\theta'_v} \right) \\ & \cdot \left(\gamma^k \cdot x\left(c_{\mathcal{S}_t}\right) - x\left(c_{\mathcal{S}_t}\right) \right). \end{aligned} \tag{15}$$

If $c_{\mathcal{S}_t}$ is nonterminal, c_R is set as

$$c_{R_t} + \gamma c_R = \Sigma_{i=0}^{k-1} \gamma^i \cdot r_{t+i} + \gamma^k \cdot x\left(c_{\mathcal{S}_t}\right) \cdot c_{\theta'_v} + \gamma \cdot x\left(c_{\mathcal{S}_t}\right) \cdot \theta'_v. \tag{16}$$

The cloud server computes

$$\begin{aligned} c_{\frac{\partial f_\pi(\theta')}{\partial \theta'}} = {} & \left(f\left(c_{\mathcal{A}_t} \mid c_{\mathcal{S}_t}\right) - \Sigma_{j=0}^{t_{\max}} f\left(c_{\mathcal{A}_t} \mid c_{\mathcal{S}_j}\right) \right) \\ & \cdot \left(\Sigma_{i=0}^{k-1} \gamma^i \cdot r_{t+i} + \gamma^k \cdot x\left(c_{\mathcal{S}_t}\right) \cdot \theta'_v + \gamma \cdot x\left(c_{\mathcal{S}_t}\right) \right. \\ & \left. \cdot \theta'_v - x\left(c_{\mathcal{S}_t}\right) \cdot \theta_v \right) + 2\beta \cdot \theta' f\left(c_{\mathcal{A}_t} \mid c_{\mathcal{S}_t}\right) \\ & \cdot \left(f\left(c_{\mathcal{A}_t} \mid c_{\mathcal{S}_t}\right) - \Sigma_{j=0}^{t_{\max}} f\left(c_{\mathcal{A}_t} \mid c_{\mathcal{S}_j}\right) \right), \end{aligned} \tag{17}$$

where $c_{\partial f_\pi(\theta')/\partial\theta'}$ is the ciphertext of $\partial f_\pi(\theta')/\partial\theta'$. The cloud server computes

$$\begin{aligned} c_{\frac{\partial f_v(\theta'_v)}{\partial \theta'_v}} = {} & 2\left(\Sigma_{i=0}^{k-1} \gamma^i \cdot r_{t+i} + \gamma^k \cdot x\left(c_{\mathcal{S}_t}\right) \cdot \theta'_v + \gamma \cdot x\left(c_{\mathcal{S}_t}\right) \right. \\ & \left. \cdot \theta'_v - x\left(c_{\mathcal{S}_t}\right) \theta'_v \right) \cdot \left(\gamma^k \cdot x\left(c_{\mathcal{S}_t}\right) + \gamma \cdot x\left(c_{\mathcal{S}_t}\right) - x\left(c_{\mathcal{S}_t}\right) \right), \end{aligned} \tag{18}$$

where $c_{\partial f_v(\theta'_v)/\partial\theta'_v}$ is the ciphertext of $\partial f_v(\theta'_v)/\partial\theta'_v$. Then, $c_{d\theta}$ is set as $c_{d\theta} + c_{\partial f_\pi(\theta')/\partial\theta'}$. $c_{d\theta_v}$ is set as $c_{d\theta_v} + c_{\partial f_\pi(\theta'_v)/\partial\theta'_v}$. The cloud server computes $c_g = \alpha c_g + (1-\alpha)(c_{d\theta})^2$ and $c_{g_v} = \alpha c_{g_v} + (1-\alpha)(c_{d\theta_v})^2$. The cloud server sends $c_{d\theta}$ and $c_{d\theta_v}$ to the user. The user decrypts $c_{d\theta}$ and $c_{d\theta_v}$ to obtain $d\theta$ and $d\theta_v$. $d\theta$ and $d\theta_v$ are encrypted as $c'_{d\theta}$ and $c'_{d\theta_v}$. $c'_{d\theta}$ and $c'_{d\theta_v}$ are returned to the cloud server. In order to reduce the depth of homomorphic multiplication for the calculation of c_g and c_{g_v}, the cloud server sends c_g and c_{g_v} to the user. The user decrypts c_g and c_{g_v} to obtain g and g_v. g and g_v are encrypted as c'_g and c'_{g_v}. The user sends c'_g and c'_{g_v} to the cloud server. The cloud server sets $c_g = c'_g$ and $c_{g_v} = c'_{g_v}$. Finally, based on the above homomorphic reciprocal of square root protocol, θ and θ_v can be updated by using equations $c_\theta = \theta - \eta(c'_{d\theta}/\sqrt{c_g + \varepsilon})$ and $c'_{\theta_v} = \theta_v - \eta(c_{d\theta_v}/\sqrt{c_{g_v} + \varepsilon})$, respectively. After the execution of Algorithm 7, we can get the encrypted optimized parameters c_θ and c_{θ_v}, which are returned to the user. The user decrypts c_θ and c_{θ_v} to obtain θ and θ_v, which can be used the implementation of secure treatment decision-making algorithm.

Table 2: Symbols and initial values.

Symbol	Initial value
θ, θ_v	0.5
θ', θ'_v	0.5
$T_{\max}, t_{\max}$	1
t_0	1
k	1
γ	0.6
r_1, r_2	0.5
β	0.1
W	1
g, g_v	0
α	0.5
ε	0.01
η	0.1

In order to better understand Algorithm 7, we give an example, which is described as follows. In this example, as shown in Table 2, we set the initial values of related parameters. We suppose that $(c_{\mathcal{S}_0},\cdots,c_{\mathcal{S}_4})$ are ciphertexts of $\mathcal{S} = (\mathcal{S}_0,\cdots,\mathcal{S}_4) = (0.1,0.1,0.15,0.3,0.35)$, respectively. $(c_{\mathcal{A}_0},\cdots,c_{\mathcal{A}_4})$ are ciphertexts of $\mathcal{A} = (\mathcal{A}_0,\cdots,\mathcal{A}_4) = (0.1,0.1,0.15,0.3,0.35)$, respectively. For the convenience of computation, we let $x(c_{\mathcal{S}_j}) = c_{\mathcal{S}_j}$, $f(c_{\mathcal{A}_j} \mid c_{\mathcal{S}_j}) = c_{\mathcal{A}_j} c_{\mathcal{S}_j}$, $\pi(\mathcal{A}_j \mid \mathcal{S}_j) = e^{f(\mathcal{A}_j \mid \mathcal{S}_j)}$, $i = 0, 1, 2, 3, 4$, $j = 0, 1, 2, 3, 4$. The cloud server first computes $c_{\pi_0} = e^{f(c_{\mathcal{A}_0} \mid c_{\mathcal{S}_0})} \cdot \theta'$ and $c_{\pi_1} = e^{f(c_{\mathcal{A}_1} \mid c_{\mathcal{S}_1})} \cdot \theta'$, where the ciphetext of c_{π_0} is $e^{0.1\times0.1\times0.5} = e^{0.005}$ and the cyphertext of c_{π_1} is $e^{0.1\times0.1\times0.5} = e^{0.005}$. Based on the implementation of the protocol $\operatorname{argmax}(c_{\pi_0}, c_{\pi_1})$, $\mathcal{A}_1$ is executed. The cloud server computes $c_{R_t} = r_t + \gamma x(c_{\mathcal{S}_t})\theta_v$, where

$$R_t = r_t + \gamma x(\mathcal{S}_t)\theta_v = 0.5 + 0.6 \times 0.1 \times 0.5 = 0.505. \tag{19}$$

Set $t = 2$. Because $\mathcal{S}_2$ is nonterminal, the cloud server computes $c_R = x(c_{\mathcal{S}_2}) \cdot \theta'_v$, where

$$R = x(\mathcal{S}_2) \cdot \theta'_v = 0.15 \times 0.5 = 0.075. \tag{20}$$

Then, the cloud server computes

$$c_R = r_2 + \gamma \cdot x\left(c_{\mathcal{S}_2}\right) \cdot \theta'_v + \gamma \cdot x\left(c_{\mathcal{S}_2}\right) \cdot \theta'_v, \tag{21}$$

where

$$R = r_2 + \gamma \cdot x(\mathcal{S}_2) \cdot \theta'_v + \gamma \cdot x(\mathcal{S}_2) \cdot \theta'_v = 0.5 + 0.6 \times 0.15 \times 0.5 + 0.6 \times 0.15 \times 0.5 = 0.59. \tag{22}$$

The cloud server computes

$$c_{\partial f_\pi\left(\theta'\right)/\partial\theta'} = \left(f\left(c_{\mathcal{A}_2} \mid c_{\mathcal{S}_2}\right) - \Sigma_{j=0}^{1} f\left(c_{\mathcal{A}_j} \mid c_{\mathcal{S}_j}\right)\right)\left(r_2 + \gamma \cdot x\left(c_{\mathcal{S}_2}\right) \cdot \theta'_v + \gamma \cdot x\left(c_{\mathcal{S}_2}\right) \cdot \theta'_v - x\left(c_{\mathcal{S}_2}\right)\theta_v\right) + 2\beta \cdot \theta' \cdot f\left(c_{\mathcal{A}_2} \mid c_{\mathcal{S}_2}\right) \cdot \left(f\left(c_{\mathcal{A}_2} \mid c_{\mathcal{S}_2}\right) - \Sigma_{j=0}^{1} f\left(c_{\mathcal{A}_j} \mid c_{\mathcal{S}_j}\right)\right), \tag{23}$$

where

$$\begin{aligned}\frac{\partial f_\pi\left(\theta'\right)}{\partial\theta'} &= \left(f(\mathcal{A}_2 \mid \mathcal{S}_2) - \Sigma_{j=0}^{1} f\left(\mathcal{A}_j \mid \mathcal{S}_j\right)\right)(r_2 + \gamma \cdot x(\mathcal{S}_2) \cdot \theta'_v + \gamma \cdot x(\mathcal{S}_2) \cdot \theta'_v - x(\mathcal{S}_2)\theta_v) \\ &\quad + 2\beta \cdot \theta' \cdot f(\mathcal{A}_2 \mid \mathcal{S}_2) \cdot \left(f(\mathcal{A}_2 \mid \mathcal{S}_2) - \Sigma_{j=0}^{1} f(\mathcal{A}_j \mid \mathcal{S}_j)\right) \\ &= (0.15 \times 0.15 - (0.1 \times 0.1 + 0.1 \times 0.1)) \cdot (0.5 + 0.6 \times 0.15 \times 0.5 + 0.6 \times 0.15 \times 0.5 - 0.15 \times 0.5) \\ &\quad + 2 \times 0.1 \times 0.5 \times 0.15 \times 0.15(0.15 \times 0.15 - (0.1 \times 0.1 + 0.1 \times 0.1)) = 0.0013.\end{aligned} \tag{24}$$

The cloud server computes

$$c_{\partial f_v(\theta'_v)/\partial\theta'_v} = 2\left(r_2 + \gamma \cdot x\left(c_{\mathcal{S}_2}\right) \cdot c_{\theta'_v} + \gamma \cdot x\left(c_{\mathcal{S}_2}\right) \cdot \theta'_v - x\left(c_{\mathcal{S}_2}\right)\theta'_v\right) \cdot \left(\gamma \cdot x\left(c_{\mathcal{S}_2}\right) + \gamma \cdot x\left(c_{\mathcal{S}_2}\right) - x\left(c_{\mathcal{S}_2}\right)\right), \tag{25}$$

where

$$\begin{aligned}\frac{\partial f_v\left(\theta'_v\right)}{\partial\theta'_v} &= 2\left(r_2 + \gamma \cdot x(\mathcal{S}_2) \cdot \theta'_v + \gamma \cdot x(\mathcal{S}_2) \cdot \theta'_v - x(\mathcal{S}_2)\theta'_v\right) \cdot (\gamma \cdot x(\mathcal{S}_2) + \gamma \cdot x(\mathcal{S}_2) - x(\mathcal{S}_2)) \\ &= 2(0.5 + 0.6 \times 0.15 \times 0.5 + 0.6 \times 0.15 \times 0.5 - 0.15 \times 0.5)(0.6 \times 0.15 + 0.6 \times 0.15 - 0.15) = 0.0309.\end{aligned} \tag{26}$$

Set $c_{d\theta} = 0$, $c_{d\theta_v} = 0$. The cloud server computes $c_{d\theta} = c_{d\theta} + c_{\partial f_\pi(\theta')}/\partial\theta'$, where

$$d\theta = d\theta + \frac{\partial f_\pi\left(\theta'\right)}{\partial\theta'} = 0.0013. \tag{27}$$

We compute $c_{d\theta_v} = c_{d\theta_v} + c_{\partial f_\pi(\theta'_v)/\partial\theta'_v}$, where

$$d\theta_v = d\theta_v + \frac{\partial f_\pi\left(\theta'_v\right)}{\partial\theta'_v} = 0.0309. \tag{28}$$

With the help of the user, the cloud server gets refreshed ciphertexts $c_{d\theta_v}$ and $c_{d\theta_v}$. The cloud server computes $c_g = \alpha c_g + (1-\alpha)(c_{d\theta})^2$, where

$$\mathrm{g} = \alpha\mathrm{g} + (1-\alpha)(\mathrm{d}\theta)^2 = (1-0.5) \times 0.0013^2 = 0.000000845. \tag{29}$$

The cloud server computes $c_{g_v} = \alpha c_{g_v} + (1-\alpha)(c_{d\theta_v})^2$, where

$$g_v = \alpha g_v + (1-\alpha)(d\theta_v)^2 = (1-0.5) \times 0.0309^2 = 0.000477405. \tag{30}$$

After the cloud server obtains ciphertexts c'_g and c'_{g_v}, we set $c_g = c'_g$ and $c_{g_v} = c'_{g_v}$. The cloud server computes $c_\theta = \theta - \eta(c'_{d\theta}/\sqrt{c_g + \varepsilon})$, where

$$\theta = \theta - \eta\frac{\mathrm{d}\theta}{\sqrt{\mathrm{g} + \varepsilon}} = 0.5 - 0.1\frac{0.0013}{\sqrt{0.000000845 + 0.01}} \approx 0.4987. \tag{31}$$

The cloud server computes $c_{\theta_v} = \theta_v - \eta(c'_{d\theta_v}/\sqrt{c_{g_v} + \varepsilon})$, where

$$\theta_v = \theta_v - \eta\frac{d\theta_v}{\sqrt{g_v + \varepsilon}} = 0.5 - 0.1\frac{0.0309}{\sqrt{0.000477405 + 0.01}} \approx 0.4698. \tag{32}$$

The user can obtain refreshed $\theta \approx 0.4987$ and $\theta_v \approx 0.4698$ by using his secret key.

6.2. Secure Treatment Decision-Making Algorithm. In this subsection, based on the above privacy-preserving A3C reinforcement learning algorithm, secure treatment decision-making algorithm $\mathrm{TDM}(\theta, \theta_v, c_x, c_{\mathcal{S}}, c_{\mathcal{A}})$ is implemented in Algorithm 8, which is described as follows. In this algorithm, input parameters include θ, θ_v, and the undiagnosed patient's encrypted current state c_x, $c_{\mathcal{S}}$, and $c_{\mathcal{A}}$. Set the index $\mathrm{col} = 0$. The ciphertext $c_{i,j}$ is initiated, where $i = 0, \cdots, \chi - 1$ and $j = 0, \cdots, \varphi - 1$. Firstly, $c_{\mathcal{S}}$'s element $c_{\mathcal{S}_j}$ is compared with c_x by using the above homomorphic comparison protocol

TDM$(\theta, \theta_v, c_x, c_{\mathcal{S}}, c_{\mathcal{A}})$:
Input: $\theta, \theta_v, c_x, c_{\mathcal{S}}, c_{\mathcal{A}}$.
Output: c_{dr}.
For $i = 0$ to $\chi - 1$.
  For $j = 0$ to $\varphi - 1$.
  Initiate $c_{i,j}$.
  End for
End for
Initiate c_b.
For $j = 0$ to $\varphi - 1$.
  $c_b = \mathrm{comp}(c_{\mathcal{S}_j}, c_x)$.
End for
Decrypt c_b to obtain b.
If $b = t$.
Set col $= j$.
$c_v = V(c_{\mathcal{S}_{\mathrm{col}}}; \theta_v) = x(c_{\mathcal{S}_j})\theta_v$.
Set $c_{\mathrm{hor}} = c_0$.
For $i = 0$ to $\chi - 1$.
  $c_{\pi,i} = \pi(c_{\mathcal{A}_i} \mid c_{\mathcal{S}_{\mathrm{col}}}) = e^{f(c_{\mathcal{A}_i} | c_{\mathcal{S}_{\mathrm{col}}})\theta} / \sum_{j=0}^{t_{\max}} e^{f(c_{\mathcal{A}_j} | c_{\mathcal{S}_j})\theta}$.
  $c_{i,\mathrm{col}} = c_v \cdot c_{\pi,i}$.
End for
Set index $= 0$.
$c_{\mathrm{hor,col}} = \mathrm{argmax}(c_{0,\mathrm{col}}, \cdots, c_{\chi-1,\mathrm{col}})$.
Set index $=$ hor.
$c_{dr} = c_{\mathcal{A}_{\mathrm{index}}}$.

Algorithm 8: Secure treatment decision-making algorithm.

Table 3: The distribution of the encrypted probability.

Encrypted probability		Encrypted actions			
		$c_{\mathcal{A}_0}$	$c_{\mathcal{A}_1}$	$c_{\mathcal{A}_2}$	$c_{\mathcal{A}_3}$
Encrypted states	$c_{\mathcal{S}_0}$	$c_{0,0}$	$c_{1,0}$	$c_{2,0}$	$c_{3,0}$
	$c_{\mathcal{S}_1}$	$c_{0,1}$	$c_{1,1}$	$c_{2,1}$	$c_{3,1}$
	$c_{\mathcal{S}_2}$	$c_{0,2}$	$c_{1,2}$	$c_{2,2}$	$c_{3,2}$
	$c_{\mathcal{S}_3}$	$c_{0,3}$	$c_{1,3}$	$c_{2,3}$	$c_{3,3}$

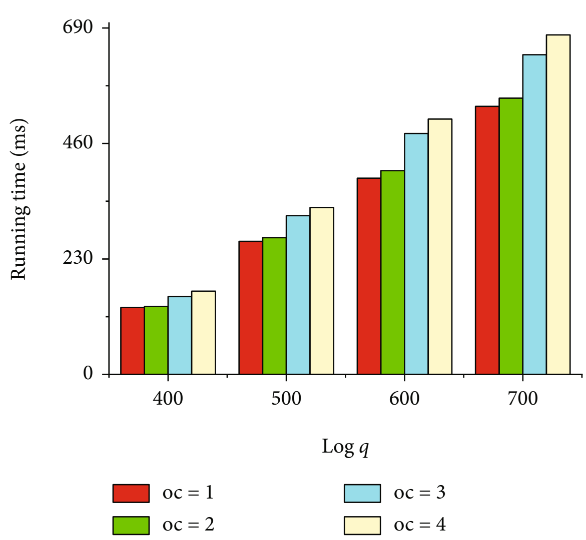

Figure 2: The efficiency of our homomorphic comparison protocol.

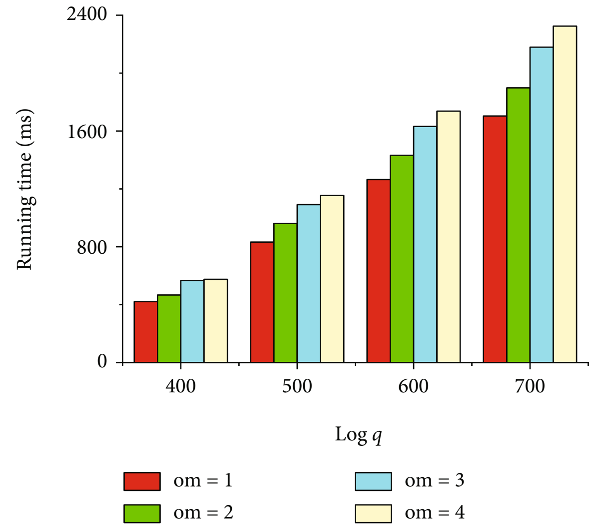

Figure 3: The efficiency of our homomorphic maximum protocol ($k = 5$).

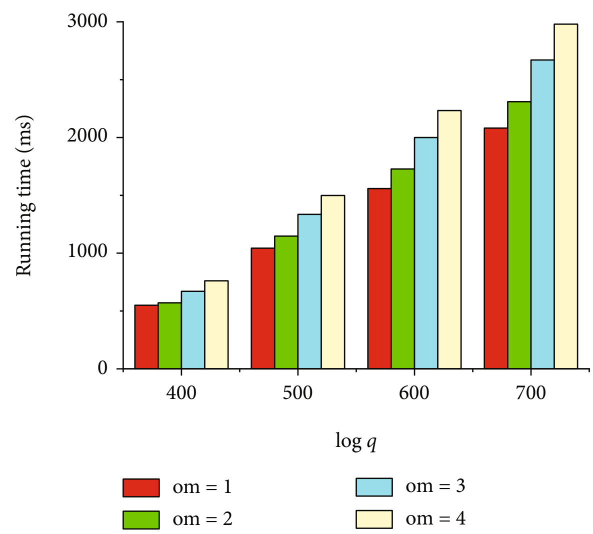

Figure 4: The efficiency of our homomorphic maximum protocol ($k = 6$).

$\mathrm{comp}(c_{\mathcal{S}_j}, c_x)$. c_b is the encrypted comparison result. After the decryption of c_b, if the comparison result $b = t$, it means that $\mathcal{S}_j = x$, set col $= j$, where $j = 0, \cdots, \varphi - 1$. Next, the cloud server computes the value function's encrypted value $c_v = V(c_{\mathcal{S}_{\mathrm{col}}}; \theta_v) = x(c_{\mathcal{S}_j}) \cdot \theta_v$, where the plaintext of c_v is $V(\mathcal{S}_{\mathrm{col}}; \theta_v)$. Set the ciphertext $c_{\mathrm{hor}} = c_0$. The cloud server computes the policy function's encrypted value

$$c_{\pi,i} = \pi\left(c_{\mathcal{A}_i} \mid c_{\mathcal{S}_{\mathrm{col}}}\right) = \frac{e^{f\left(c_{\mathcal{A}_i} | c_{\mathcal{S}_{\mathrm{col}}}\right)\theta}}{\sum_{j=0}^{t_{\max}} e^{f\left(c_{\mathcal{A}_j} | c_{\mathcal{S}_j}\right)\theta}}, \quad (33)$$

where the plaintext of $c_{\pi,i}$ is $\pi(\mathcal{A}_i \mid \mathcal{S}_{\mathrm{col}})$. The cloud server computes homomorphic multiplication between c_v and $c_{\pi,i}$, namely, $c_{i,\mathrm{col}} = c_v \cdot c_{\pi,i}$, where $c_{i,\mathrm{col}}$ is the

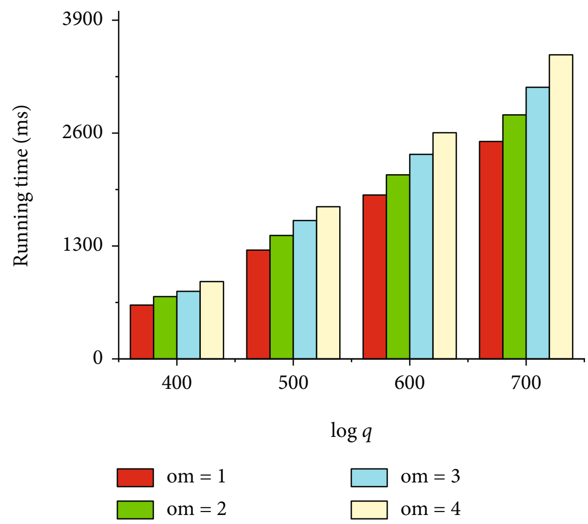

Figure 5: The efficiency of our homomorphic maximum protocol ($k=7$).

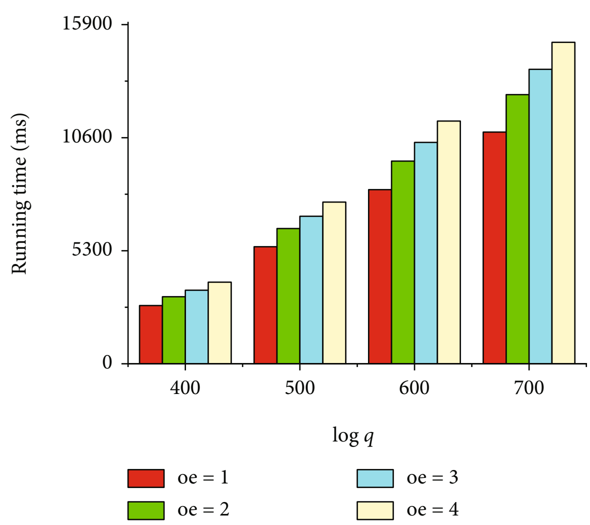

Figure 6: The efficiency of our homomorphic exponential protocol ($n=2$).

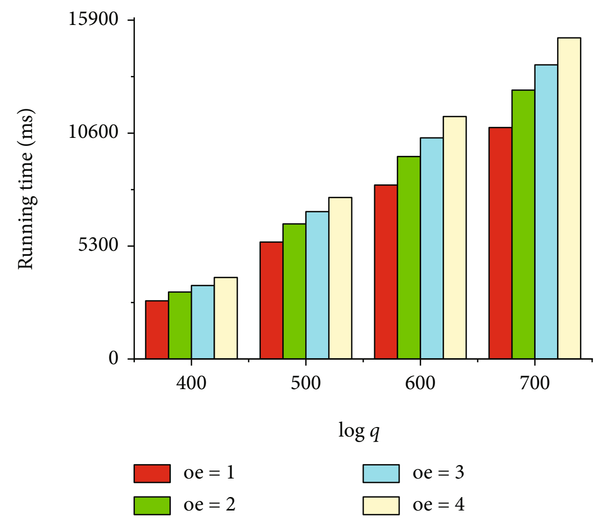

Figure 7: The efficiency of our homomorphic exponential protocol ($n=3$).

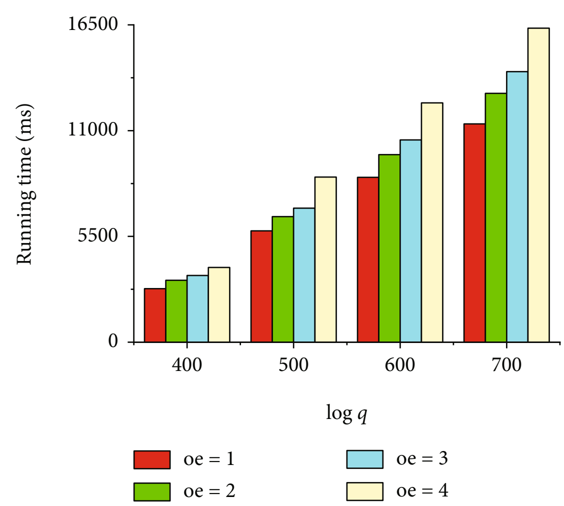

Figure 8: The efficiency of our homomorphic exponential protocol ($n=4$).

ciphertext of $V(\mathcal{S}_{\mathrm{col}};\theta_v)\pi(\mathcal{A}_i \mid \mathcal{S}_{\mathrm{col}})$, $i=0,\cdots,\chi-1$. Set index $=0$. Finally, the cloud server computes the ciphertext $c_{\mathrm{hor,col}}$ by using the homomorphic maximum protocol $\mathrm{argmax}(c_0,\cdots,c_{\chi-1})$. Set index $=$ hor. Hence, the treatment decision is $c_{dr}=c_{\mathcal{A}_{\mathrm{index}}}$. Then, c_{dr} will be transmitted to the undiagnosed patient. In order to obtain the treatment decision dr, c_{dr} can be decrypted by using his own secret key.

In order to better understand Algorithm 8, we give an example, which is described as follows. In this example, we set $\chi=4$, $\varphi=4$, $\theta=0.5$, $\theta_v=0.5$, and $t_{\max}=3$. We suppose that $(c_{\mathcal{S}_0},\cdots,c_{\mathcal{S}_3})$ are ciphertexts of (0.1,0.2,0.3,0.4), respectively. $(c_{\mathcal{A}_0},\cdots,c_{\mathcal{A}_3})$ are ciphertexts of (0.1,0.2,0.3,0.4), respectively. For the convenience of computation, we let $x(c_{\mathcal{S}_j})=c_{\mathcal{S}_j}$, $f(c_{\mathcal{A}_i} \mid c_{\mathcal{S}_j})=c_{\mathcal{A}_i}c_{\mathcal{S}_j}$, $\pi(\mathcal{A}_i \mid \mathcal{S}_j)=e^{f(\mathcal{A}_i \mid \mathcal{S}_j)}$, $i=0,1,2,3$, $j=0,1,2,3$. The calculation of $c_{i,j}=V(c_{\mathcal{S}_j};\theta_v)\pi(c_{\mathcal{A}_i} \mid c_{\mathcal{S}_j})$ is the key component for the execution of Algorithm 8, where the corresponding plaintext of $c_{i,j}$ is $V(\mathcal{S}_j;\theta_v)\pi(\mathcal{A}_i \mid \mathcal{S}_j)$. In this example, the distribution of $c_{i,j}$ can be shown in Table 3.

If the patient requires a diagnostic service, the encrypted current state c_x is input for the implementation of Algorithm 8. Then, $c_{\mathcal{S}_{be}}$ is compared with c_x by the execution of the protocol $c_b=\mathrm{comp}(c_{\mathcal{S}_{be}},c_x)$, where $be=0$. If the comparison result $b=t$, set col $=be$. If the comparison result $b\neq t$, the next element $c_{\mathcal{S}_{be+1}}$ will be compared with c_x until $be+1>3$. We suppose that $x=\mathcal{S}_1$, namely, col $=1$. Hence, $c_v=x(c_{\mathcal{S}_1})\theta_v=x(c_{\mathcal{S}_1})0.5$, where the plaintext of c_v is $0.2\times0.5=0.1$. We compute ciphertexts $c_{\pi,0}$, $c_{\pi,1}$, $c_{\pi,2}$, and $c_{\pi,3}$, which corresponding plaintexts are $e^{0.01}$/temp, $e^{0.02}$/temp, $e^{0.03}$/temp, and $e^{0.04}$/temp, respectively, where temp $=e^{0.01}+e^{0.02}+e^{0.03}+e^{0.04}$. Then, we compute ciphertexts $c_{0,1}$, $c_{1,1}$, $c_{2,1}$,

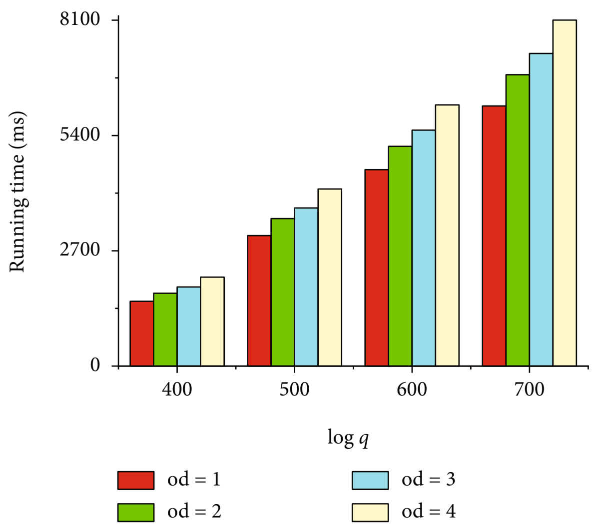

FIGURE 9: The efficiency of our homomorphic division protocol.

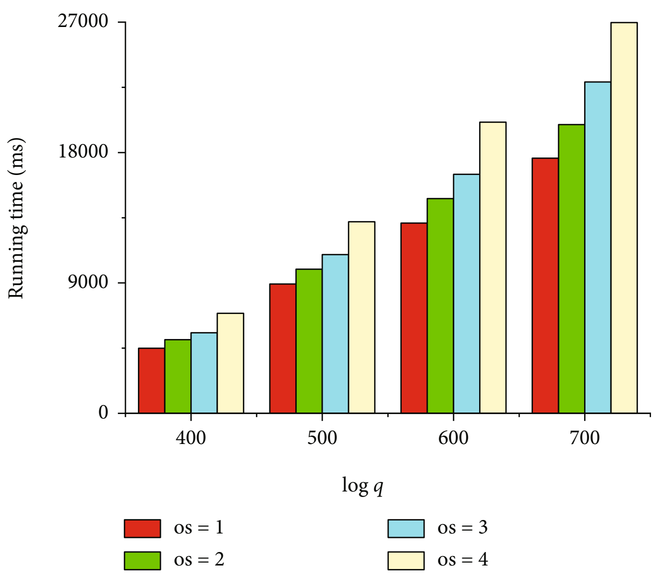

FIGURE 10: The efficiency of our homomorphic reciprocal of square root protocol.

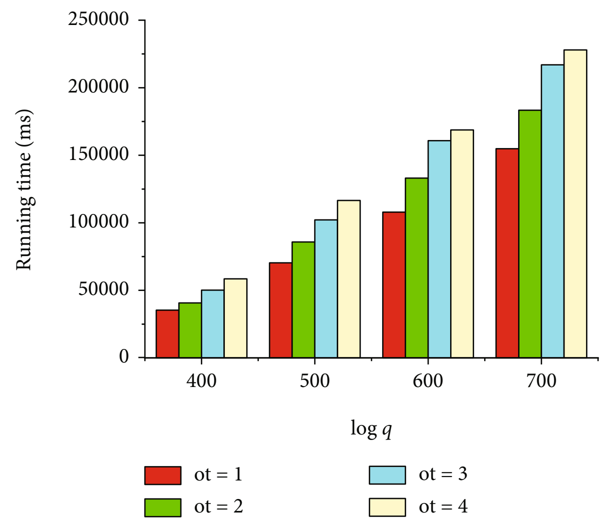

FIGURE 11: The efficiency of our secure A3C reinforcement learning algorithm.

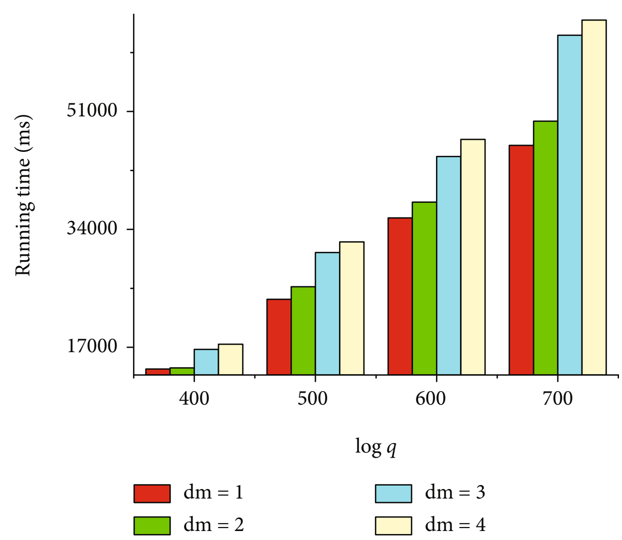

FIGURE 12: The efficiency of our secure treatment decision-making algorithm.

and $c_{3,1}$, which corresponding plaintexts are $0.1e^{0.01}$/temp, $0.1e^{0.02}$/temp, $0.1e^{0.03}$/temp, and $0.1e^{0.04}$/temp, respectively. Based on the execution of the protocol argmax$(c_{0,1}, c_{1,1}, c_{2,1}, c_{3,1})$, we can obtain the output ciphertext is $c_{3,1}$. The encrypted treatment decision is $c_{\mathscr{A}_3}$, which corresponding plaintext is $\mathscr{A}_3$.

7. Performance Results

In this section, based on Cheon et al.'s homomorphic encryption scheme, we analyze the efficiency of our secure computation protocols, secure A3C reinforcement learning algorithm, and secure treatment decision-making algorithm. We use the virtual machine to implement experiments without the GPU hardware platform. In our experimental environment, the operating system is macOS 10.14.6. Our personal computer has two Intel (R) Core (TM) i5 CPU processors, which runs at 2.3 GHz with 8.00 GB RAM. The operation system of a virtual machine is ubuntu 16.04. The virtual machine is allocated single Intel (R) Core (TM) i5 CPU processor with 1.0 GB RAM. In order to implement high-level numeric algorithms, we choose the NTL library. We use the GCC platform to compile our C++ codes. We adopt the UC Irvine Machine Learning Repository (http://archive.ics.uci.edu/ml/index.php) for implementing the experiments. For convenience, we set $\log q$ ranging from 400 to 700, the scaling factor $\log p = 30$.

Figure 2 shows the efficiency of our homomorphic comparison protocol, where the number of comparison oc ranges from 1 to 4. As shown in Figure 2, the running time of our homomorphic comparison protocol increases significantly with the increasing of oc. Figures 3–5 show the efficiency of our homomorphic maximum protocol, where the number of maximum *om* ranges from 1 to 4 and the number of plaintexts k ranges from 5 to 7. It can be easily observed that the running time of our homomorphic maximum protocol increases significantly with the increasing of k and *om*. Figures 6–8 show the efficiency of our homomorphic exponential protocol, where the number of exponential operation *oe* ranges from 1 to 4 and the integer n ranges from 2 to 4. We can observe that the running time of our homomorphic exponential protocol increases rapidly with the increasing of *oe* and n.

Then, Figure 9 shows the efficiency of our homomorphic division protocol, where the number of division *od* ranges from 1 to 4. We can observe the changing trend of the running time of our homomorphic division protocol. This protocol has an obvious growth of running time with the increasing of *od* and $\log q$. Figure 10 shows the efficiency of our homomorphic reciprocal of square root protocol, where *os* ranges from 1 to 4; *os* denotes the number of operations of reciprocal of square root. With the increasing of *os* and $\log q$, more running time is needed for implementing our homomorphic reciprocal of square root protocol. It can be observed that its running time is longer than the above homomorphic comparison, maximum, exponential, and division protocols. Figure 11 shows the efficiency of our secure A3C reinforcement learning algorithm, where *ot* ranges from 1 to 4; *ot* denotes the number of operations of A3C training algorithm. With the increasing of *ot* and $\log q$, our A3C reinforcement learning algorithm requires more running time. This algorithm is responsible for training the parameters θ and θ_v. Hence, this algorithm is complicated. We can observe too much running time is needed for this algorithm, which can demonstrate the above viewpoint. Figure 12 shows the efficiency of our secure treatment decision-making algorithm, where *dm* ranges from 1 to 4; *dm* denotes the number of operations of treatment decision-making algorithm. The running time of this algorithm grows with the increasing of *dm*. This algorithm uses the optimized θ and θ_v. Hence, this algorithm is less complicated than the secure A3C algorithm. The running time of this algorithm is shorter than the secure A3C algorithm, which can verify the above viewpoint. In a conclusion, the above efficiency analysis shows the feasibility of our secure computation protocols and algorithms.

8. Conclusion

Reinforcement learning is helpful for implementing dynamic treatment regimes on health data. However, private health data may be illegally leaked, falsified, or deleted in the execution of the reinforcement learning algorithm. Hence, we study secure dynamic treatment regimes on health data. In this paper, we have designed homomorphic comparison protocol, homomorphic maximum protocol, homomorphic exponential protocol, homomorphic division protocol, and homomorphic reciprocal of square root protocol. Based on these secure computation protocols, we have proposed a privacy-preserving A3C reinforcement learning algorithm for the first time. Then, it is used for implementing the secure treatment decision-making algorithm. Finally, we simulate the proposed secure computation protocols and algorithms. Simulation results show that our secure computation protocols and algorithms are feasible.

In the future research, we will use homomorphic encryption to implement other machine learning algorithms, such as distributed learning [37] and federated reinforcement learning [38], which can successfully dominate multiple real devices that have the same type and slightly different dynamics. In addition, we plan to evaluate the performance of the secure A3C algorithm in other real-world scenarios, for example, vehicular ad hoc network.

Data Availability

The data of secure computation protocols and algorithms used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Science and Technology Innovation Projects of Shenzhen (JCYJ20190809152003992), Shenzhen Science and Technology Program (JCYJ20210324100813034), the Guangdong Basic and Applied Basic Research Foundation (2020A1515110496), and the College-Enterprise Collaboration Project of Shenzhen Institute of Information Technology (11400-2021-010201-010199).

References

[1] I. M. Tayler and R. S. Stowers, "Engineering hydrogels for personalized disease modeling and regenerative medicine," *Acta Biomaterialia*, vol. 132, pp. 4–22, 2021.

[2] X. Liu, R. Deng, K. K. Raymond Choo, and Y. Yang, "Privacy-preserving reinforcement learning design for patient-centric dynamic treatment regimes," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 1, pp. 456–470, 2021.

[3] Y. Liu, B. Logan, N. Liu, Z. Xu, J. Tang, and Y. Wang, "Deep reinforcement learning for dynamic treatment regimes on medical registry data," in *Proceedings of 2017 IEEE International Conference on Healthcare Informatics*, pp. 380–385, Park City, UT, USA, 2017.

[4] R. S. Sutton and A. G. Barto, "Reinforcement learning," *A Bradford Book*, vol. 15, no. 7, pp. 665–685, 1998.

[5] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: a survey," *Mobile Networks and Applications*, vol. 26, no. 3, pp. 1145–1168, 2021.

[6] X. Sun, F. R. Yu, and P. Zhang, "A survey on cyber-security of connected and autonomous vehicles (CAVs)," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–20, 2021.

[7] T. Yang, L. Kong, N. Zhao, and R. Sun, "Efficient energy and delay tradeoff for vessel communications in SDN based maritime wireless networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3800–3812, 2021.

[8] T. Yang, M. Qin, N. Cheng, W. Xu, and L. Zhao, "Liquid Software-Based Edge Intelligence for Future 6G Networks," *IEEE Network*, 2021.

[9] T. Yang, J. Chen, and N. Zhang, "AI-empowered maritime internet of things: a parallel-network-driven approach," *IEEE Network*, vol. 34, no. 5, pp. 54–59, 2020.

[10] L. Liu, J. Feng, Q. Pei et al., "Blockchain-enabled secure data sharing scheme in mobile-edge computing: an asynchronous advantage actor–critic learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2342–2353, 2020.

[11] X. Sun, F. R. Yu, P. Zhang, Z. Sun, W. Xie, and X. Peng, "A survey on zero-knowledge proof in blockchain," *IEEE Network*, vol. 35, no. 4, pp. 198–205, 2021.

[12] S. Mao, J. Wu, L. Liu, D. Lan, and A. Taherkordi, "Energy-efficient cooperative communication and computation for wireless powered mobile-edge computing," *IEEE Systems Journal*, pp. 1–12, 2020.

[13] C. Cimpanu, "Amca data breach has now gone over the 20 million mark," 2019, https://www.zdnet.com/article/amca-data-breach-has-nowgone-over-the-20-million-mark/.

[14] W. Zhang, M. Li, R. Tandon, and H. Li, "Online location trace privacy: an information theoretic approach," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 1, pp. 235–250, 2019.

[15] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of Secure Computation*, vol. 4, no. 11, pp. 169–180, 1978.

[16] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Proceedings of Advances in Cryptology - ASIACRYPT 2017*, pp. 409–437, Hong Kong, China, 2017.

[17] H. Kim and W. Lee, "Real-time path planning through Q-learning's exploration strategy adjustment," in *Proceedings of 2021 International Conference on Electronics, Information, and Communication*, pp. 1–3, Jeju, Republic of Korea, 2021.

[18] C. Wu, Z. Liu, F. Liu, T. Yoshinaga, Y. Ji, and J. Li, "Collaborative learning of communication routes in edge-enabled multi-access vehicular environment," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 4, pp. 1155–1165, 2020.

[19] R. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[20] V. Mnih, A. Badia, M. Mirza et al., "Asynchronous methods for deep reinforcement learning," in *Proceedings of the 33rd International Conference on Machine Learning*, pp. 1928–1937, New York, USA, 2016.

[21] J. Feng, F. Richard Yu, Q. Pei, X. Chu, J. du, and L. Zhu, "Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: a deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6214–6228, 2020.

[22] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing - STOC '09*, pp. 169–178, New York, USA, 2009.

[23] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based," in *Advances in Cryptology – CRYPTO 2013*, pp. 75–92, Springer, 2013.

[24] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory*, vol. 6, no. 3, pp. 1–36, 2014.

[25] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Advances in Cryptology – EUROCRYPT 2010*, pp. 1–23, Springer, 2010.

[26] A. Khedr and G. Gulak, "Securemed: secure medical computation using gpu-accelerated homomorphic encryption scheme," *IEEE Journal of Biomedical and Health Informatics*, vol. 22, no. 2, pp. 597–606, 2018.

[27] X. Sun, P. Zhang, M. Sookhak, J. Yu, and W. Xie, "Utilizing fully homomorphic encryption to implement secure medical computation in smart cities," *Personal and Ubiquitous Computing*, vol. 21, no. 5, pp. 831–839, 2017.

[28] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Manual for using homomorphic encryption for bioinformatics," *Proceedings of the IEEE*, vol. 105, no. 3, pp. 1–16;, 2017.

[29] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Proceedings of Theory of Cryptography Conference*, pp. 325–341, Cambridge, USA, 2005.

[30] A. Poon, S. Jankly, and T. Chen, "Privacy preserving Fishers exact test on genomic data," in *Proceedings of 2018 IEEE International Conference on Big Data*, pp. 2546–2553, Seattle, USA, 2018.

[31] J. L. Raisaro, G. Choi, S. Pradervand et al., "Protecting privacy and security of genomic data in i2b2 with homomorphic encryption and differential privacy," *IEEE/ACM Transactions on Computational Biology & Bioinformatics*, vol. 15, no. 5, pp. 1413–1426, 2018.

[32] X. Liu, R. H. Deng, K.-K. R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2401–2414, 2016.

[33] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," 2012, https://eprint.iacr.org/2012/144.pdf.

[34] L. Jiang, L. Chen, T. Giannetsos, B. Luo, K. Liang, and J. Han, "Toward practical privacy-preserving processing over encrypted data in IoT: an assistive healthcare use case," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10177–10190, 2019.

[35] X. Sun, P. Zhang, J. K. Liu, J. Yu, and W. Xie, "Private machine learning classification based on fully homomorphic encryption," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 352–364, 2020.

[36] C. Lomont, "Fast inverse square root," 2003, http://lomont.org/papers/2003/InvSqrt.pdf.

[37] X. Chen, C. Wu, Z. Liu, N. Zhang, and Y. Ji, "Computation offloading in beyond 5g networks: a distributed learning framework and applications," *IEEE Wireless Communications*, vol. 28, no. 2, pp. 56–62, 2021.

[38] H.-K. Lim, J.-B. Kim, C.-M. Kim, G.-Y. Hwang, H.-b. Choi, and Y.-H. Han, "Federated reinforcement learning for controlling multiple rotary inverted pendulums in edge computing environments," in *Proceedings of 2020 International Conference on Artificial Intelligence in Information and Communication*, pp. 463-464, Durban, South Africa, 2020.