Privacy-Preserving Patient-Centric Clinical Decision Support System on Naïve Bayesian Classification

Ximeng Liu, *Student Member, IEEE*, Rongxing Lu, *Member, IEEE*,
Jianfeng Ma, Le Chen, and Baodong Qin

***Abstract*—Clinical decision support system, which uses advanced data mining techniques to help clinician make proper decisions, has received considerable attention recently. The advantages of clinical decision support system include not only improving diagnosis accuracy but also reducing diagnosis time. Specifically, with large amounts of clinical data generated everyday, naïve Bayesian classification can be utilized to excavate valuable information to improve a clinical decision support system. Although the clinical decision support system is quite promising, the flourish of the system still faces many challenges including information security and privacy concerns. In this paper, we propose a new privacy-preserving patient-centric clinical decision support system, which helps clinician complementary to diagnose the risk of patients' disease in a privacy-preserving way. In the proposed system, the past patients' historical data are stored in cloud and can be used to train the naïve Bayesian classifier without leaking any individual patient medical data, and then the trained classifier can be applied to compute the disease risk for new coming patients and also allow these patients to retrieve the top-k disease names according to their own preferences. Specifically, to protect the privacy of past patients' historical data, a new cryptographic tool called additive homomorphic proxy aggregation scheme is designed. Moreover, to leverage the leakage of naïve Bayesian classifier, we introduce a privacy-preserving top-k disease names retrieval protocol in our system. Detailed privacy analysis ensures that patient's information is private and will not be leaked out during the disease diagnosis phase. In addition, performance evaluation via extensive simulations also demonstrates that our system can efficiently calculate patient's disease risk with high accuracy in a privacy-preserving way.**

***Index Terms*—Clinical decision support system (CDSS), naïve Bayesian classifier, patient centric, privacy preserving.**

Manuscript received August 18, 2014; revised December 25, 2014; accepted February 17, 2015. Date of publication February 24, 2015; date of current version March 3, 2016. This work was supported by the Key Program of NSFC-Guangdong Union Foundation under Grant U1135002; the Key Program of NSFC under Grant U1405255; the National Natural Science Foundation of China under Grant 61402109 and Grant 61370078; the support of Nanyang Technological University under Grant NTU-SUG (M4081196) and MOE Tier 1 (M4011177).

X. Liu is with the School of Telecommunication Engineering, Xidian University, Xi'an 710071, China, and also with the School of Electrical and Electronics Engineering, Nanyang Technological University, 639798 Singapore (e-mail: snbnix@gmail.com).

R. Lu and L. Chen are with the School of Electrical and Electronics Engineering, Nanyang Technological University, 639798 Singapore (e-mail: rxlu@ntu.edu.sg; chenle0213@gmail.com).

J. Ma is with the School of Computer Science and Technology, Xidian University, Xi'an 710071, China (e-mail: jfma@mail.xidian.edu.cn).

B. Qin is with School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: qinbaodong@sjtu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JBHI.2015.2407157

I. Introduction

HEALTHCARE industry, extensively distributed in the global scope to provide health services for patients, has never faced such a massive amounts of electronic data or experienced such a sharp growth rate of data today. As stated by the Institute for Health Technology Transformation (iHT^2), U.S. health care data alone reached 150 exabytes (10^{18} bytes) in 2011 and would soon reach zettabyte (10^{21} bytes) scale and even yottabytes (10^{24} bytes) in the future [1]. However, if no appropriate technique is developed to find great potential economic values from big healthcare data, these data might not only become meaningless but also require a large amount of space to store and manage. Over the past two decades, the miraculous evolution of data mining technique has imposed a major impact on the revolution of human's lifestyle by predicting behaviors and future trends on everything, which can convert stored data into meaningful information. These techniques are well suitable for providing decision support in the healthcare setting. To speed up the diagnosis time and improve the diagnosis accuracy, a new system in healthcare industry should be workable to provide a much cheaper and faster way for diagnosis. Clinical decision support system (CDSS), with various data mining techniques being applied to assist physicians in diagnosing patient diseases with similar symptoms, has received a great attention recently [2]–[4]. Naïve Bayesian classifier, one of the popular machine learning tools, has been widely used recently to predict various diseases in CDSS [27]. Despite its simplicity, it is more appropriate for medical diagnosis in healthcare than some sophisticated techniques [6], [7].

The CDSS with naïve Bayesian classifier has offered many advantages over the traditional healthcare systems and opens a new way for clinicians to predict patient's diseases. However, its flourish still hinges on understanding and managing the information security and privacy challenges, especially during the patient disease decision phase. One of the main challenges is how to keep patient's medical data away from unauthorized disclosure. The usage of medical data can be of interest for a large variety of healthcare stakeholders. For example, an online direct-to-consumer service provider offers individual risk prediction for patient's disease. Without good protection of patient's medical data, patient may feel afraid that his medical data will be leaked and abused, and refuse to provide his medical data to CDSS for diagnosis. Therefore, it is crucial to protect patient's medical data. However, keeping medical data's privacy are not sufficient to push forward the whole CDSS into flourish. Service provider's classifier, which is used to predict patient's disease, cannot be exposed to third parties since the classifier is considered as service provider's own asset. Otherwise, the third parties can abuse the classifier to predict patient's disease, which could damage service provider's profit. Therefore, besides preserving the privacy of patient's medical data, how to protect service provider's privacy is also crucial for the CDSS.

In this paper, to address the privacy issues lying in the CDSS, we propose a privacy-preserving patient-centric clinical decision support system, called PPCD, which is based on naïve Bayesian classification to help physician to predict disease risks of patients in a privacy-preserving way. Specifically, the main contributions of this paper are fourfold.

First, we propose a secure and PPCD, which allows service provider to diagnose patient's disease without leaking any patient's medical data. In PPCD, the past patient's historical medical data can be used by service provider to train the naïve Bayesian classifier. Then, service provider can use the trained classifier to diagnose patient's diseases according to his symptoms in a privacy-preserving way. Finally, patients can retrieve the diagnosed results according to his own preference privately without compromising the service provider's privacy.

Second, since individual historical medical data will disclose patient's sensitive medical data to service provider, to minimize patient's privacy disclosure, we also introduce a new aggregation technique called additive homomorphic proxy aggregation (AHPA) scheme, which allows service provider to build naïve Bayesian classifier without leaking any individual historical medical data. Even the service provider and cloud platform (CP) collude, no party can get any information about the individual historical medical data except for the owner himself, and only the aggregated data can be accessed by the service provider.

Third, to cater for patient-centric diagnosis result retrieval, we present a privacy-preserving top-k disease names retrieval protocol (*TOP-K*), which allows patients to retrieve diagnosed disease names according to their preferences. Within the *TOP-K*, service provider learns nothing about patient's symptom information used for diagnosis and patient also cannot get anything about naïve Bayesian classifier. Note that although privacy-preserving top-k protocol was studied in [8], yet both computation cost and communication overhead about privacy-preserving top-k protocol in [8] are expensive, which make it unsuitable for PPCD. Compared with the privacy-preserving top-k protocol, our proposed *TOP-K* is much more efficient in terms of computation cost and communication overhead.

Finally, to validate the efficiency of the proposed PPCD, we also develop a custom simulator built in Java. Extensive simulation demonstrates that our PPCD can efficiently help patient to diagnose the disease with high predict success rate and it also minimizes privacy disclosure without overburdening the whole system.

The remainder of this paper is organized as follows: In Section II, we describe some preliminaries, which serve as the basis of our PPCD. In Section III, we formalize the system model, the privacy requirements, and identify our design goal. In Section IV, we present our PPCD, followed by privacy analysis and performance evaluation in Section V and Section VI, respectively. In Section VII, we also discuss some related works. Finally, we draw our conclusions in Section VIII.

TABLE I
DEFINITIONS AND NOTATIONS IN PPCD

Symbol	Definition
PK_c	User c' public key of Paillier encryption scheme
SK_c	User c' private key of Paillier encryption scheme
pk_a	User a' public key of *AHPR* scheme
sk_a	User a' private key of *AHPR* scheme
$E_{PK_c}(\cdot)$	Paillier encryption function
$D_{SK_c}(\cdot)$	Paillier decryption function
$[\cdot]_{pk_P}$	Ciphertext of *AHPR* scheme
$C_1, C_2, \ldots, C_{n_d}$	n_d disease classes in the system
$A_1, A_2, \ldots, A_{n_s}$	n_s symptom attributes in the system
$X_j^{(i)}$	j-th symptom of patient i
$Y_j^{(i)}$	j-th disease of patient i
ID_{C_i}	Disease name of class C_i
$P(C_i = 1)$	The probability for getting disease C_i
$P(C_i = 0)$	The probability for not getting disease C_i
$\vec{\mathbf{X}}^{(i)}$	Symptom vector of patient i
$\vec{\mathbf{Y}}^{(i)}$	Disease vector of patient i
$\vec{\mathbf{X}}_{pk_P}^{(i)}$	Encrypted symptom vector by public key pk_P
$T_i^{(c)}$	Encrypted tuple $\langle E_{pk_c}(P(C_i)), E_{pk_c}(ID_{C_i})\rangle$
$a \cdot b$	Multiplication between a and b over cyclic group

II. Preliminaries

In this section, we outline the naïve Bayesian classifier [9], Paillier homomorphic encryption [10], and secure multiplication (*SM*) protocol [11], which will serve as the basis of the proposed PPCD. Before that we first use Table I to list main notations used in the whole paper.

A. Naïve Bayesian Classifier

Naïve Bayesian classifier [9] is a very attractive classifier, which has been proved to be effective in many practical applications, including text classification [12], [13], medical diagnosis [14], [15], and systems performance management [16]. Here, we briefly review the naïve Bayesian classifier as follows.

There are f classes which are denoted as $C_1, C_2, \ldots, C_f$. Each sample is represented by n-dimensional vector $\vec{\mathbf{X}} = \{X_1, \ldots, X_n\}$, depicting n measured values of the n attributes $A_1, \ldots, A_n$, respectively. The classifier needs to predict $\vec{\mathbf{X}}$ belongs to the class with the highest a posteriori probability, i.e., $\vec{\mathbf{X}}$ is predicted to lie in the class C_i if and only if there exists i, such that

$$P(C_i|\vec{\mathbf{X}}) > P(C_j|\vec{\mathbf{X}}), \quad \text{for all } 1 \leq j \leq f, j \neq i.$$

By Bayes's theorem

$$P(C_i|\vec{\mathbf{X}}) = \frac{P(\vec{\mathbf{X}}|C_i)P(C_i)}{P(\vec{\mathbf{X}})}$$

we can see that $P(\vec{\mathbf{X}})$ is the same for all classes, only $P(\vec{\mathbf{X}}|C_i)P(C_i)$ needs to be maximized. In order to evaluate $P(\vec{\mathbf{X}}|C_i)P(C_i)$, the naive assumption on class conditional independence is made. This presumes that the values of all attributes are conditionally independent of one another, mathematically meaning

$$P(\vec{\mathbf{X}}|C_i) \approx \prod_{k=1}^{n} P(X_k|C_i).$$

The probabilities $P(X_1|C_i), P(X_2|C_i), \ldots, P(X_n|C_i)$ can easily be estimated from the training set. In all, the classifier predicts that the class label of $\vec{\mathbf{X}}$ is C_i if and only C_i has maximized the value $P(\vec{\mathbf{X}}|C_i)P(C_i)$ among $P(\vec{\mathbf{X}}|C_j)P(C_j), (1 \leq j \leq f)$.

B. Paillier Homomorphic Encryption

In order to achieve PPCD, we will adopt Paillier homomorphic encryption [10] as one of the building blocks. We briefly review the steps involved in Paillier homomorphic encryption as follows.

Key Generation: Given the security parameter k and two large prime numbers p and q, where $|p| = |q| = k$, compute $N = pq$ and $\lambda = lcm(p-1, q-1)$. Define a function $L(x) = \frac{x-1}{N}$, then choose a generator $g \in \mathbb{Z}^*_{N^2}$ and calculate $\mu = (L(g^\lambda \bmod N^2))^{-1}$. Then, the public key is denoted as $pk = (N, g)$ and the corresponding private key is $sk = (\lambda, \mu)$.

Encryption: Given a message $m \in \mathbb{Z}_N$, choose a random number $r \in \mathbb{Z}^*_N$. The ciphertext can be calculate as $I = E_{pk}(m) = g^m \cdot r^N \bmod N^2$.

Decryption: Given a ciphertext I, the message can be recovered from the ciphertext by calculating $m = D_{sk}(I) = L(I^\lambda \bmod N^2) \cdot \mu \bmod N$.

Assume both $E_{pk}(x) = g^x \cdot r_1^N \bmod N^2$ and $E_{pk}(y) = g^y \cdot r_2^N \bmod N^2$ are encrypted under the same public key pk. The Paillier encryption has the following properties.

1) *Additive Homomorphism:* Given two ciphertexts $E_{pk}(x)$ and $E_{pk}(y)$, it has $D_{sk}(E_{pk}(x) \cdot E_{pk}(y)) = D_{sk}(g^x r_1^N \cdot g^y r_2^N \bmod N^2 = g^{x+y} r_1^N r_2^N \bmod N^2) = x + y$.
2) *Scalar-Multiplicative Homomorphism:* Given constant number $c \in \mathbb{Z}_N$ and a ciphertext $E_{pk}(x)$, it has $D_{sk}(E_{pk}(x)^c) = D_{sk}(g^{cx}(r_1^c)^N \bmod N^2) = c \cdot x$.
3) *Self-Blinding:* Given a ciphertext $E_{pk}(x)$, it is efficient to recover the plaintext of the ciphertext by calculating $D_{sk}(E_{pk}(x) \cdot (r')^N) = D_{sk}(g^x(r_1 r')^N \bmod N^2) = x$. Note that for the given $x \in \mathbb{Z}_N$, $E_{pk}(-x) = E_{pk}(x)^{N-1}$.

C. SM Protocol

Because Paillier encryption only supports additive homomorphism, which cannot achieve multiplication of the plaintext, we also use *SM* protocol described in [11] as a building block to design our PPCD. Two parties (called Alice and Bob) will be involved in this protocol for execution. Consider Bob has two encrypted data $E_{pk}(x)$ and $E_{pk}(y)$, Alice has a secret key sk. If $E_{pk}(x)$ and $E_{pk}(y)$ are directly sent to Alice for decryption, it will damage the other party's interests (see Section IV-C for detailed explanation). The goal of *SM* protocol is to calculate $E_{pk}(x \cdot y)$ without leaking x and y to each other. The overall steps of *SM* protocol are shown as follows.

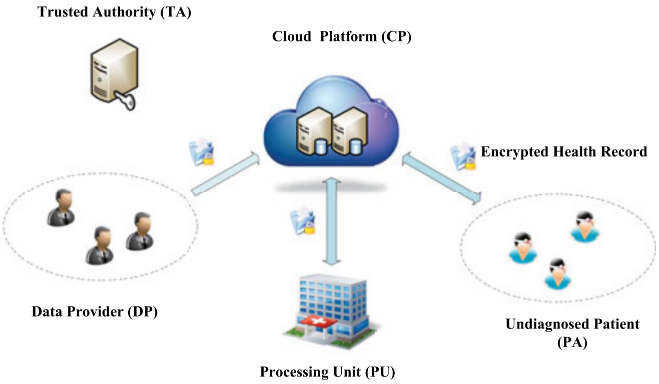


Fig. 1. System model under consideration.

Step 1: Bob selects two random numbers $r_x, r_y \in \mathbb{Z}_N$, calculates $x' = E_{pk}(x) \cdot E_{pk}(r_x)$ and $y' = E_{pk}(y) \cdot E_{pk}(r_y)$, sends x' and y' to Alice.

Step 2: Alice decrypts x' and y' by using the secret key sk, and multiplies them as $h = D_{sk}(x') \cdot D_{sk}(y')$. Then, Alice encrypts h by using pk and denotes it as $h' = E_{pk}(h)$, and sends h' to Bob. It can be easily verified that $h = (x + r_x)(y + r_y)$.

Step 3: Bob first calculates $s_1 = E_{pk}(x)^{N-r_y}$, $s_2 = E_{pk}(y)^{N-r_x}$, and $s_3 = E_{pk}(r_x \cdot r_y)^{N-1}$. Then, Bob calculates the following formula to gain the encrypted $x \cdot y$:

$$h' \cdot s_1 \cdot s_2 \cdot s_3 = E_{pk}(h - r_y \cdot x - r_x \cdot y - r_x \cdot r_y) = E_{pk}(x \cdot y).$$

III. System Model, Privacy Requirements, and Design Goal

In this section, we formalize the system model, privacy requirements, and identity our design goals.

A. System Model

In our system, we mainly focus on how to securely train naïve Bayesian classifier and use the classifier to clinical decide patients' disease without leaking their private information. Specifically, we define the system model by dividing PPCD into five parties: Trusted authority (TA), Cloud Platform (CP), data provider (DP), processing unit (PU), and undiagnosed patient (PA). The overall system model of our PPCD can be found in Fig. 1.

1) Trusted Authority: TA is the indispensable entity, which is trusted by all entities involved in the system, who is in charge of distributing and managing all the private keys involve in the system.

2) Cloud Platform: CP contains unlimited storage space, which can store and manage all the data in the system. Other parties who have limited storage space can outsource their data to CP for storing. In addition, CP has some computation abilities to perform some calculations over the stored data.

3) Data Provider: DP can provide historical medical data that contain patients' symptoms and confirmed diseases, which are used for training naïve Bayesian classifier. All these data are outsourced to CP for storing.

4) Processing Unit (PU, Representing as Service Provider in the Previous Section): PU can be a company or hospital, which can provide online direct-to-consumer service and offer individual risk prediction for various diseases based on client's symptoms. PU uses historical medical data to construct naïve Bayesian classifier and then use the model to predict the disease risk of undiagnosed patients.

5) Undiagnosed Patient (PA): PA has some symptom information, which is collected during doctor visits or directly provided by patient (e.g., blood pressure, heart rate, weight, etc). The symptoms can be sent to PU for diseases diagnosis.

B. Privacy Requirements

Privacy is crucial for the success of patient's diseases diagnosis. In our privacy model, we consider DP is trustable, which provides correct historical medical data. The internal party PU is considered as *curious-but-honest,* which is interested in DP's individual historical medical data and PA's medical data, but strictly follows the protocols executed in the system. Both CP and PA are also *curious-but-honest* parties in this system. PA is curious about PU's classifier, while CP is curious about all the other parties' data in the system. Moreover, an external adversary is interested in all data transmitted in the system by eavesdropping. Therefore, in order to prevent both internal party from information leakage and external adversary from eavesdropping, the following privacy requirements should be satisfied in PPCD.

1) *DP's Privacy:* DP's historical medical data contain confirmed case records of patient's symptoms and confirmed diseases. These historical medical data can be used to train naïve Bayesian classifier. These individual data contain some sensitive information, which are highly related to patient's privacy. It cannot be directly exposed to untrust parties during the transmission and storage. Otherwise, DP will not provide its own data to the other parties due to the privacy information leakage. Therefore, privacy of DP should be preserved in our system.
2) *PU's Privacy:* PU uses historical medical data to train naïve Bayesian classifier and gets conditional probabilities about the classifier. These probabilities are considered as an asset of PU, which cannot directly be sent to patients or leaked to other parties during the disease diagnosis.
3) *PA's Privacy:* PA contains some symptom data, which are sensitive and cannot directly expose to other parities. In addition, the diagnosis results are also highly sensitive information, which cannot be leaked to other parties. If needed, PA can let the authorized person (authorized clinician) disclose the diagnosis results for further processing.

C. Design Goals

In order to achieve the secure medical decision for UP under the aforementioned model, our system design will fulfill privacy and performance guarantees as follows.

1) *The Proposed System Should Achieve Privacy-Preserving Requirements:* As stated above, if CDSS does not consider the privacy requirements, patient's highly sensitive information (symptom and disease information) will be disclosed to PU, CP, and unauthorized parties in the patient's medical decision. It will let patient unwillingly provide its own data to CDSS. In addition, PU is always a profit company, which prevents his own data from leaking to other parties in the system. Therefore, the proposed system should achieve the privacy of PA and PU simultaneously.

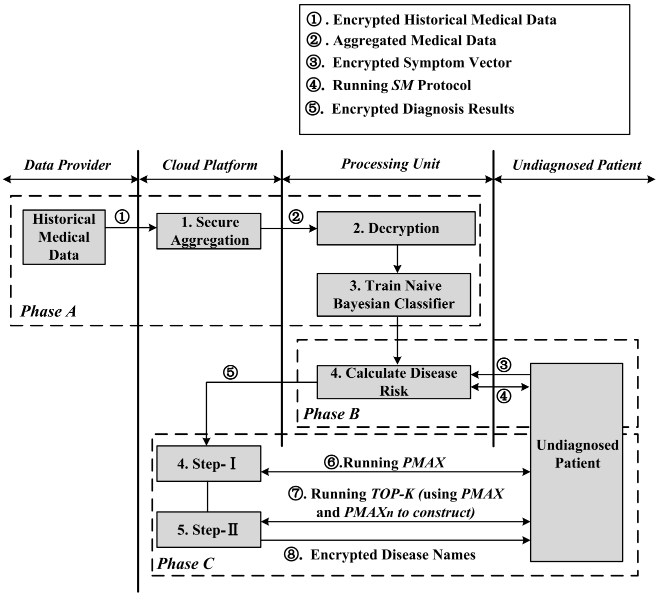

①. Encrypted Historical Medical Data
②. Aggregated Medical Data
③. Encrypted Symptom Vector
④. Running SM Protocol
⑤. Encrypted Diagnosis Results
Data Provider
Cloud Platform
Processing Unit
Undiagnosed Patient
Historical Medical Data
①
1. Secure Aggregation
②
2. Decryption
3. Train Naive Bayesian Classifier
Phase A
⑤
4. Calculate Disease Risk
③
④
Phase B
Undiagnosed Patient
4. Step-Ⅰ
⑥.Running PMAX
⑦. Running TOP-K (using PMAX and PMAXn to construct)
5. Step-Ⅱ
⑧. Encrypted Disease Names
Phase C

Fig. 2. Overall procedure of PPCD.

2) *The Proposed System Should Achieve Computation Efficiency:* The patient always has limited computational resources, which cannot support overburden computation. To support patient-centric diagnosis results retrieval from CP in time, the proposed system should consider computation efficiency. Therefore, it is important to allow PA to retrieve diagnosis results in realtime.

IV. PROPOSED PPCD SYSTEM

In this section, we present our PPCD system for predicting patient's disease risk, which mainly consists of the following three phases: privacy-preserving training naïve Bayesian classifier (*Phase A*), privacy-preserving computing patient of disease risk (*Phase B*), privacy-preserving retrieving top-k diagnosed results (*Phase C*). The overall procedure of PPCD is listed in Fig. 2.

A. Privacy-Preserving Training Naïve Bayesian Classifier

In this phase, DP should provide his historical medical data to PU for training naïve Bayesian classifier and these data should be sent to CP for storage. We first construct a new cryptographic tool called *AHPA* scheme to securely aggregate the message to solve the collusion problem between PU and CP. Then, we use the tool to train naïve Bayesian classifier privately.

1) AHPA Scheme: Our *AHPA* scheme is based on ElGamal encryption scheme [17], [18] and contains the following six algorithms: *KeyGen*, *Rekeygen*, *Encrypt*, *Decrypt*, *Reencrypt&Agg*, and *Redecrypt*.

a) KeyGen: The algorithm is run by TA to generate public key and private key for DP i and PU. Choose two cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order N. Let g be a generator of group $\mathbb{G}$ and compute $X = e(g, g)$. For each DP i, chooses $a_{i,1}, a_{i,2} \in \mathbb{Z}_N$, and computes $e(g, g)^{a_{i,1}}$ and $g^{a_{i,2}}$. Denote DP i's public key as $pk_{a_i} = \{e(g, g)^{a_{i,1}}, g^{a_{i,2}}\}$ and private key $sk_{a_i} = \{a_{i,1}, a_{i,2}\}$, respectively. PU's public key and private key can be computed in the same way and be denoted as $pk_P = \{e(g, g)^{p_1}, g^{p_2}\}$ and $sk_P = \{p_1, p_2\}$, respectively.

b) Rekeygen: The algorithm is run by TA to generate the reencryption key. This key can be generated by DP i's private key a_i, PU's public key g^{p_2}, and a random number r'_i selected by TA, i.e., $sk_{a_i \to P} = (g^{p_2})^{a_{i,1}} g^{r'_i} = g^{a_{i,1} p_2} g^{r'_i}$. Moreover, TA generates a private R for PU to allow PU to decrypt the aggregated message. R is generated by using r'_i, DP's random number g^{r_i}, and PU's private key p_2, where

$$R = e(g, g)^{\sum r_i r'_i p_2^{-1}}.$$

c) Encrypt: This algorithm is executed by DP i. Let $x^{(i)} \in \mathbb{Z}_N$ be the message, which can be encrypted under DP i's public key pk_{a_i}. Then, the ciphertext can be calculated as $[x^{(i)}]_{pk_{a_i}} = \{A_{i,1}, A_{i,2}\} = \{g^{r_i}, e(g, g)^{x^{(i)}} \cdot e(g, g)^{a_{i,1} r_i}\}$, where r_i is a random number from $\mathbb{Z}_N$.

d) Decrypt: $[x^{(i)}]_{pk_{a_i}}$ can be decrypted by using DP i's private key sk_{a_i}

$$\frac{A_{i,2}}{e(g, A_{i,1})^{a_{i,1}}} = \frac{e(g, g)^{x^{(i)}} \cdot e(g, g)^{a_{i,1} r_i}}{e(g, g^{r_i})^{a_{i,1}}} = e(g, g)^{x^{(i)}}.$$

Similar to other works in [19] and [20], $x^{(i)}$ is a finite and relatively small number, DP can compute discrete logarithm of $e(g, g)^{x^{(i)}}$ in base $e(g, g)$ to retrieve $x^{(i)}$.

e) Reencrypt&Agg: This algorithm is executed by CP. The algorithm can be processed as follows.

1) For each DP i, the ciphertext in DP i's domain $[x^{(i)}]_{pk_{a_i}}$ can be reencrypted into PU's domain by $sk_{a_i \to P}$. That is, the ciphertext $[x^{(i)}]_{pk_{a_i}}$ can be converted into $[x^{(i)}]_{pk_p} = \{B_{i,1}, B_{i,2}\} = \{e(g, g)^{a_{i,1} p_2 r_i} e(g, g)^{r'_i r_i}, e(g, g)^{x^{(i)}} \cdot e(g, g)^{a_{i,1} r_i}\}$, where $B_{i,1}$ can be computed by $e(sk_{a_i \to p}, A_1) = e(g^{a_{i,1} p_2} g^{r'_i}, g^{r_i}) = e(g, g)^{a_{i,1} p_2 r_i} e(g, g)^{r'_i r_i}$.
2) For all $i = 1, \ldots, l$, CP calculates the aggregated ciphertext by using $[x^{(i)}]_{pk_p}$

$$CT_{\text{Agg}} = \{CT_1, CT_2\},$$

where

$$CT_1 = \prod_{i=1}^{l} B_{i,1} = e(g, g)^{p_2 \sum_{i=1}^{l} a_{i,1} r_i} e(g, g)^{\sum_{i=1}^{l} r_i r'_i}$$

$$CT_2 = \prod_{i=1}^{l} B_{i,2} = e(g, g)^{\sum_{i=1}^{l} x^{(i)}} \cdot e(g, g)^{\sum_{i=1}^{l} a_{i,1} r_i}$$

l is the number of DP.

f) Redecrypt: This algorithm is executed by PU. PU can decrypt the aggregated ciphertext CT_{Agg} by using sk_P

$$\frac{CT_2 \cdot R}{(CT_1)^{p_2^{-1}}} = \frac{e(g, g)^{\sum x^{(i)}} \cdot e(g, g)^{\sum a_{i,1} r_i} e(g, g)^{\sum r_i r'_i p_2^{-1}}}{e(g, g)^{\sum a_{i,1} r_i} e(g, g)^{\sum r_i r'_i p_2^{-1}}}$$

$$= e(g, g)^{\sum_{i=1}^{l} x^{(i)}}.$$

$\sum_{i=1}^{l} x^{(i)}$ is a finite and relatively small number, which can be obtained by calculating discrete logarithm.

2) Privacy-Preserving Training Naïve Bayesian Classifier: In this section, we show how to train naïve Bayesian classifier privately by using *AHPA*.

Let $\vec{\mathbf{X}}^{(i)} = (X_1^{(i)}, \ldots, X_{n_s}^{(i)})$ represent patient i's symptom vector, while $\vec{\mathbf{Y}}^{(i)} = (Y_1^{(i)}, \ldots, Y_{n_d}^{(i)})$ denotes i's corresponding disease vector, where n_s is the number of symptom categories involved in the system, n_d is the number of disease categories involved in the system. In these historical data, if patient i has the jth symptom, then $X_j^{(i)} = 1$, and $X_j^{(i)} = 0$ otherwise. If patient i has tth disease, then $Y_t^{(i)} = 1$, and $Y_t^{(i)} = 0$ otherwise. For each symptom, each DP uses his own public key pk_i to encrypt the historical symptom $[X_j^{(i)}]_{pk_i}$ and confirmed diseases $[Y_t^{(i)}]_{pk_i}$, and then sends them to CP to store. Note that all the symptoms should be normalized into binary before encryption, that is, $X_j^{(i)}$ should be converted to $(X_1'^{(i)}, \ldots, X_{n'}'^{(i)})$ instead, where $X_j^{(i)}$ is numeric, $X_1'^{(i)}, \ldots, X_{n'}'^{(i)}$ are binary, n' is the value range of $X_j^{(i)}$. For example, the attribute temperature X ranges from 35.5 to 41.5 °C, which should be converted into $(X_1, \ldots, X_{51})$. If DP i's $X^{(i)} = 35.5$ °C, then the converted attributes $X_1'^{(i)} = 1$, and $X_j'^{(i)} = 0$, where $j = 2 \ldots 51$.

Once all the historical medical data have been outsourced, CP first uses *Reencrypt&Agg* algorithm to transfer the ciphertext from DP's domain $[X_j^{(i)}]_{pk_i}$ into CP's domain $[X_j^{(i)}]_{pk_p}$, then aggregates the encrypted symptom data $\vec{\mathbf{X}}_{pk_p}^{(i)} = ([X_1^{(i)}]_{pk_p}, \ldots, [X_{n_s}^{(i)}]_{pk_p})$ into one encrypted vector $\vec{\mathbf{X}}'_{pk_p} = ([X'_1]_{pk_p}, \ldots, [X'_{n_s}]_{pk_p})$ by calculating

$$[X'_j]_{pk_P} \leftarrow \mathit{Reencrypt\&Agg}([X_j^{(1)}]_{pk_P}, \ldots, [X_j^{(l)}]_{pk_P})$$

where l is the number of the historical medical data. In addition, the aggregated disease vector $\vec{\mathbf{Y}}'_{pk_P} = ([Y'_1]_{pk_P}, \ldots, [Y'_{n_d}]_{pk_P})$ can be calculated in the same way, where

$$[Y'_j]_{pk_P} \leftarrow \mathit{Reencrypt\&Agg}([Y_j^{(1)}]_{pk_P}, \ldots, [Y_j^{(l)}]_{pk_P}).$$

After performing aggregation, CP sends $[X'_1]_{pk_P}, \ldots, [X'_{n_s}]_{pk_P}$ and $[Y'_1]_{pk_P}, \ldots, [Y'_{n_d}]_{pk_P}$ to PU. PU uses its own private key sk_P to decrypt these encrypted elements and gets the vectors $\vec{\mathbf{X}}' = (X'_1, \ldots, X'_{n_s})$ and $\vec{\mathbf{Y}}' = (Y'_1, \ldots, Y'_{n_d})$. Note that l is the total number of the historical medical data stored in the CP, then PU can calculate

$$\hat{P}(A_j = 1 | C_t = 1) = \frac{X'_j}{Y'_t}, \hat{P}(A_j = 1 | C_t = 0) = \frac{X'_j}{l - Y'_t}$$

$$\hat{P}(C_t = 1) = \frac{Y'_t}{l}, \hat{P}(C_t = 0) = \frac{l - Y'_t}{l}$$

$$\hat{P}(A_j = 0|C_t = 1) = 1 - \hat{P}(A_j = 1|C_t = 1)$$
$$\hat{P}(A_j = 0|C_t = 0) = 1 - \hat{P}(A_j = 1|C_t = 0).$$

These $\hat{P}(F_j|C_t)(j = 1, \ldots, n_s; t = 1, \ldots, n_d)$ should be kept privately by PU as its own asset. It is worth noticing that the *AHPA* scheme can only encrypt integer, while conditional probability cannot be directly used to calculate patient's disease risk. To address the issue, all the conditional probabilities should be expanded into integers by calculating $P(C_t = 1) = \lfloor C'\hat{P}(C_t = 1)\rfloor$. For example, if $\hat{P}(C_1 = 1) = 0.38$ and $\hat{P}(C_1 = 0) = 0.48$, both $\hat{P}(C_1 = 1)$ and $\hat{P}(C_1 = 0)$ can be expanded into $P(C_1 = 1) = 38$ and $P(C_1 = 0) = 48$ by using $C' = 100$. Note that this expansion will not effect the relationship between the probabilities. The probability of $\hat{P}(A_j|C_t = 1)$, $\hat{P}(A_j|C_t = 0)$ and $\hat{P}(C_t = 0)$ should be expanded in the same way by using C'.

B. Privacy-Preserving Computing Patient of Disease Risk

In the rest of the paper, all the data are correlated with a specific PA, and we denote the specific PA as c. Because (c) will always appear in the notation, without special explanation, all the data represent c's data and we will drop (c) from expression, for example, we transform $X_1^{(c)}$ into X_1 in order to keep the notation uncluttered.

Suppose c has n_s measured symptoms called $X_1, \ldots, X_{n_s}$, he encrypts these symptoms as $E_{PK_c}(X_j)(j = 1, \ldots, n_s)$ and sends them to PU via CP. Once these encrypted symptoms are received, PU cannot decrypt the c's encrypted symptoms since he has not gotten c's private key SK_c. PU can use the Bayes's theorem to calculate patient's probability of having disease (The detail description of Bayes's theorem can be found in [21]). For every diseases t, PU uses $P(A_j|C_t)$ to calculate

$$\begin{aligned} & E_{PK_c}(P(A_j = X_j|C_t = 1)) \\ = \; & E_{PK_c}(X_j P(A_j = 1|C_t = 1) \\ & + (1 - X_j)P(A_j = 0|C_t = 1)) \\ = \; & E_{PK_c}(X_j)^{P(A_j=1|C_t=1)} \\ & \cdot E_{pk_c}(1 - X_j)^{P(A_j=0|C_t=1)}. \end{aligned} \quad (1)$$

With the same method, CP can calculate $E_{PK_c}(P(A_j = X|C_t = 0))$. Next, CP calculates

$$\hat{K}_{t,1} = E_{PK_c}(\prod_{j=1}^{n_s} P(A_j = X_j|C_t = 1) \cdot P(C_t = 1)). \quad (2)$$

The calculation in (2) can be achieved by using *SM* protocol. Denote $\hat{H}_{t,1} = \prod_{j=1}^{n_s} P(A_j = X_j|C_t = 1) \cdot P(C_t = 1)$, then we have $\hat{K}_{t,1} = E_{PK_c}(\hat{H}_{t,1})$. In addition, PU uses c's public key to encrypt disease name ID_t and denotes it as

$$\hat{G}_{t,1} = E_{PK_c}(\hat{ID}_{t,1}).$$

Furthermore, PU can compute $\langle \hat{K}_{t,0}, \hat{G}_{t,0}\rangle$ in the similar way, where

$$\hat{K}_{t,0} = E_{PK_c}(\hat{H}_{t,0}), \hat{G}_{t,0} = E_{PK_c}(\hat{ID}_{t,0})$$

with $\hat{H}_{t,0} = \prod_{j=1}^{n_s} P(A_j = X_j|C_t = 0) \cdot P(C_t = 0)$ and $\hat{ID}_{t,0} = 0$. In some cases, CP may know the disease names according to the order sorting of the ciphertext even without decryption (some background knowledges). Once some encrypted disease names are retrieved, the privacy about this patient will be leaked to CP. In order to protect patient's privacy, all the ciphertexts should be permuted before outsourcing. This procedure is described as follows.

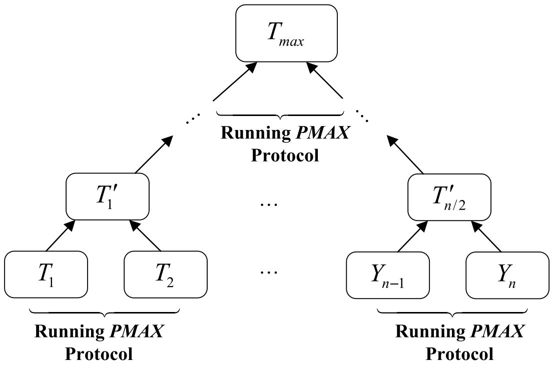


Fig. 3. Running procedure of *PMAX*$_n$.

PU first randomly permutes $\langle \hat{K}_{t,\pi_d(e)}, \hat{G}_{t,\pi_d(e)}\rangle_{e=\{0,1\}}$ by using permutation π_d. Then, PU uses permutation π to permute and denote them as $\{\hat{T}_{a,b}\} = \{\langle \hat{K}_{\pi(t),\pi_d(e)}, \hat{G}_{\pi(t),\pi_d(e)}\rangle\}_{e=\{0,1\},t=\{1,\ldots,n\}}$, and then sends them to CP for storing, where $b = \pi_d(e)$, $a = \pi(t)$.

C. Privacy-Preserving Retrieving Top-k Diagnosed Results

When these diagnosed results $\{\hat{T}_{a,b}\}_{a=\{1,\cdots,n\},b=\{0,1\}}$ have been computed, CP needs to find the encrypted top-k diagnosis results and sends them back to PA. This procedure can be divided into two steps.

Step1: CP needs to judge whether PA suffers from some specific diseases according to probabilities, that is, if $\hat{H}_{a,1} \geq \hat{H}_{a,0}$, denote $E_{PK_c}(ID_a) = E_{PK_c}(\hat{ID}_{a,1})$, otherwise, denote $E_{PK_c}(ID_a) = E_{PK_c}(\hat{ID}_{a,0})$.

Step 2: CP needs to select top-k possible disease names according to diagnosis probabilities, that is, select $E_{PK_c}(ID'_1), \ldots, E_{PK_c}(ID'_k)$ from $E_{PK_c}(ID_1), \ldots, E_{PK_c}(ID_{n_d})$ where its corresponding $H'_1, \ldots, H'_t$ are top-k probabilities among $H_1, \ldots, H_{n_d}$.

In order to finish these two steps, three protocols should be designed called Privacy-preserving maximum protocol (*PMAX*), privacy-preserving maximum out of n protocol (*PMAX*$_n$), and privacy-preserving top-k disease names retrieval protocol (*TOP-K*), respectively. CP can securely achieve *Step 1* by running *PMAX* and *Step 2* can be finished by executing *TOP-K*, where *PMAX*$_n$ is designed as basic component of *TOP-K*. Here, we describe these three protocols as follows.

1) Privacy-Preserving Maximum Protocol: PMAX allows CP to get a new ciphertext tuple $T_U = \langle E_{PK_c}(H_U), E_{PK_c}(ID_U)\rangle$ from two different ciphertext tuples $T_A = \langle E_{PK_c}(H_A), E_{PK_c}(ID_A)\rangle$ and $T_B = \langle E_{PK_c}(H_B), E_{PK_c}(ID_B)\rangle$, where $H_U = \max(H_A, H_B)$ and

```
Algorithm 1: PRIVACY-PRESERVING MAXIMUM OUT OF $n$ PROTOCOL ($PMAX_n$)
Input: CP has $n_d$ tuples $T_1, \cdots, T_{n_d}$, PA holds private key $SK_c$.
Output: the maximum tuple $T_U$ among $T_1, \cdots, T_{n_d}$.
Initialize set $S_b$ such that $S_b = \{T_1, \cdots, T_{n_d}\}$.
for $i = 1$ to $\lceil log_2 n_d \rceil$ do
    initialize $S_a$ such that $S_a = \emptyset$.
    for $j \in \lfloor \frac{|S_b|}{2} \rfloor$ do
        calculate $T'_j = PMAX(T_{2j-1}, T_{2j})$.
        add $T'_j$ to set $S_a$.
    Set $S_a = S_b$.
$S_b$ contains only one element $T_U$.
Return $T_U$.
```

ID_U correspond to disease name. By runnning *PMAX*, CP can get the new ciphertext tuple T_U; however, he cannot judge whether the origin of T_U comes from T_A or T_B. We give the specific construction of the CP as

Step 1 (@ CP): 1) CP converts $E_{PK_c}(H_A), E_{PK_c}(H_B)$ into $E_{PK_c}(H'_A), E_{PK_c}(H'_B)$ by calculating

$$\begin{aligned} E_{PK_c}(H'_A) &= E_{PK_c}(1) \cdot E_{PK_c}(H_A)^2 = E_{PK_c}(2H_A + 1) \\ E_{PK_c}(H'_B) &= E_{PK_c}(H_B)^2 = E_{PK_c}(2 \cdot H_B). \end{aligned}$$

It is worth noting that if $H_A \geq H_B$ and $H_A \geq 0$ and $H_B \geq 0$, we have $H'_A > H'_B$. 2) CP chooses random values $R, r_3, r^*, \hat{r}^* \in \mathbb{Z}_N$ and tosses a random coin $s \in \{0, 1\}$. If $s = 1$, CP computes

$$\begin{aligned} C_1 &= E_{PK_c}(H'_A)^R \cdot E_{PK_c}(H'_B)^{N-R} \cdot g^{2^L} r_3^N \bmod N^2 \\ &= E'_{PK_c}(R(H'_A - H'_B) + 2^L). \end{aligned} \tag{3}$$

$$\begin{aligned} C_2 &= E_{PK_c}(H_B) \cdot E_{PK_c}(H_A)^{N-1} \cdot E_{PK_c}(r^*) \bmod N^2 \\ &= E_{PK_c}(H_B - H_A + r^*). \end{aligned} \tag{4}$$

$$\begin{aligned} C_3 &= E_{PK_c}(ID_B) \cdot E_{PK_c}(ID_A)^{N-1} \cdot E_{PK_c}(\hat{r}^*) \bmod N^2 \\ &= E_{PK_c}(ID_B - ID_A + \hat{r}^*). \end{aligned} \tag{5}$$

If $s = 0$, CP computes:

$$\begin{aligned} C_1 &= E_{PK_c}(H'_B)^R \cdot E_{PK_c}(H'_A)^{N-R} \cdot g^{2^L} r_3^N \bmod N^2 \\ &= E'_{PK_c}(R(H'_B - H'_A) + 2^L). \end{aligned} \tag{6}$$

$$\begin{aligned} C_2 &= E_{PK_c}(H_A) \cdot E_{PK_c}(H_B)^{N-1} \cdot E_{PK_c}(r^*) \bmod N^2 \\ &= E_{PK_c}(H_A - H_B + r^*). \end{aligned} \tag{7}$$

$$\begin{aligned} C_3 &= E_{PK_c}(ID_A) \cdot E_{PK_c}(ID_B)^{N-1} \cdot E_{PK_c}(\hat{r}^*) \bmod N^2 \\ &= E_{PK_c}(ID_A - ID_B + \hat{r}^*). \end{aligned} \tag{8}$$

where $|L| = |RC'|$, C' is the expansion value which is introduced in Section IV. The goal of using 2^L is to make the value $R(H'_A - H'_B) + 2^L$ positive all the time.

After computation, CP sends C_1, C_2, C_3 to PA.

Step 2 (@ PA): Once C_1, C_2, C_3 are received, PA decrypts C_1 by using his own private key SK_c as

$$M = D_{SK_c}(C_1).$$

Then, PA will check the relationship between M and 2^L.

If $M > 2^L$, PA denotes $\alpha = 0$ and calculates $D'_2 = E_{PK_c}(0), D'_3 = E_{PK_c}(0)$.

If $M < 2^L$, PA denotes $\alpha = 1$ and calculates $D'_2 = C_2 \cdot \hat{r}_1^N, D'_3 = C_3 \cdot \hat{r}_2^N$, where $\hat{r}_1$ and $\hat{r}_2$ are randomly selected from $\mathbb{Z}_N$.

Encrypt α as $E_{PK_c}(\alpha)$. PA sends $E_{PK_c}(\alpha)$, D'_2, and D'_3 back to CP.

Step 3 (@ CP): After receiving $E_{PK_c}(\alpha)$, D'_2, and D'_3, CP will do the following calculations according to s: if $s = 1$, CP calculates

$$\begin{aligned} E_{PK_c}(H_U) &= E_{PK_c}(H_A) \cdot D'_2 \cdot E_{PK_c}(\alpha)^{N-r^*} \\ E_{PK_c}(ID_U) &= E_{PK_c}(ID_A) \cdot D'_3 \cdot E_{PK_c}(\alpha)^{N-\hat{r}^*}. \end{aligned}$$

If $s = 0$, CP calculates

$$\begin{aligned} E_{PK_c}(H_U) &= E_{PK_c}(H_B) \cdot D'_2 \cdot E_{PK_c}(\alpha)^{N-r^*} \\ E_{PK_c}(ID_U) &= E_{PK_c}(ID_B) \cdot D'_3 \cdot E_{PK_c}(\alpha)^{N-\hat{r}^*}. \end{aligned}$$

It can be easily verified that H_U is the maximum one between H_A and H_B.

The correctness of PMAX: The correctness of the protocol can be seen as follows.

If $s = 1$ and $H'_A > H'_B$, then $\alpha = 0$ and $D'_2 = E_{PK_c}(0)$ and $D'_3 = E_{PK_c}(0)$. The $E_{PK_c}(H_U)$ and $E_{PK_c}(ID_U)$ can be calculated as follows:

$$\begin{aligned} E_{PK_c}(H_U) &= E_{PK_c}(H_A) \cdot D'_2 \cdot E_{PK_c}(\alpha)^{N-r^*} \\ &= E_{PK_c}(H_A) \cdot E_{PK_c}(0) \cdot E_{PK_c}(0)^{N-r^*} \\ &= E_{PK_c}(H_A) \\ E_{PK_c}(ID_U) &= E_{PK_c}(ID_A) \cdot D'_3 \cdot E_{PK_c}(\alpha)^{N-\hat{r}^*} \\ &= E_{PK_c}(ID_A) \cdot E_{PK_c}(0) \cdot E_{PK_c}(0)^{N-\hat{r}^*} \\ &= E_{PK_c}(ID_A). \end{aligned}$$

If $s = 1$ and $H'_A < H'_B$, then $\alpha = 1$ and $D'_2 = E_{PK_c}(H_B - H_A + r^*)$ and $D'_3 = E_{PK_c}(ID_B - ID_A + \hat{r}^*)$. The $E_{PK_c}(H_U)$ and $E_{PK_c}(ID_U)$ can be calculated as follows:

$$\begin{aligned} E_{PK_c}(H_U) &= E_{PK_c}(H_A) \cdot D'_2 \cdot E_{PK_c}(\alpha)^{N-r^*} \\ &= E_{PK_c}(H_A) \cdot E_{PK_c}(H_B - H_A + r^*) \\ &\quad \cdot E_{PK_c}(1)^{N-r^*} = E_{PK_c}(H_B) \\ E_{PK_c}(ID_U) &= E_{PK_c}(ID_A) \cdot D'_3 \cdot E_{PK_c}(\alpha)^{N-\hat{r}^*} \\ &= E_{PK_c}(ID_A) \cdot E_{PK_c}(ID_B - ID_A + \hat{r}^*) \cdot \\ &\quad E_{PK_c}(1)^{N-\hat{r}^*} = E_{PK_c}(ID_B). \end{aligned}$$

When $s = 0$, $E_{PK_c}(H_U)$ and $E_{PK_c}(ID_U)$ can be calculated in the same way. From the correctness verification, we can see that the protocol can indeed get maximum tuple from the two encrypted tuples.

2) Privacy-Preserving Maximum Out of n Protocol: The goal of *PMAX*$_n$ is to allow CP to get a new ciphertext tuple T_U with maximum H_U from the n encrypted tuples $T_1, \ldots, T_n$. Neither plaintext information nor ciphertext relationship will be leaked to both CP and PA during running this protocol. The specific construction of *PMAX*$_n$ is proposed in Algorithm 1. CP first initializes the set S_b with the n encrypted tuples, where these n tuples are the input of this algorithm. The set is used to store the candidate encrypted tuples. Another set S_a will also be used to store the intermediate result of the maximum encrypted tuple between T_{2j} and T_{2j+1} from the set S_b. After comparison, S_a is assigned to S_b. This procedure will be executed until there is only one encrypted tuple leave in S_b. The tuple is with maximum probability among the n encrypted tuples. It is worth noting that the algorithm should check whether the number of tuples in S_b is odd or not before executing *PMAX*. If the cardinality of S_b is odd, the algorithm should add one extra tuple $\langle E_{PK_c}(0), E_{PK_c}(0)\rangle$ to S_b before running *PMAX*. The overall procedure of *PMAXn* is listed in Fig. 3.

3) Privacy-Preserving Top-k Disease Names Retrieval Protocol (TOP-K): We will design *TOP-K,* which allows PA to get top-k disease results from CP. The specific construction of *TOP-K* is listed in Algorithm 2. First, CP initializes a set S'_a, which is used to store intermediate result. This algorithm will run for k loops. For each loop, *PMAX*$_n$ will be executed to allow CP to get the maximum tuple T_{MAX} from set S'_a. Then, for every element that belongs to S'_a, choose random numbers $R_j \in \mathbb{Z}_N$ and calculate

$$\begin{aligned} V_j &= (E_{PK_c}(H_{\text{MAX}}) \cdot E_{PK_c}(H_j)^{N-1})^{R_j} \\ &= E_{PK_c}(R_j(H_{\text{MAX}} - H_j)). \end{aligned}$$

Then, CP uses permutation π_i to permute these V_j and denotes these permuted data as $V_{\pi_i(j)}$, and then sends them to PA. Once these tuples are received, PA will calculate $A_{\pi_i(j)}$ and $B_{\pi_i(j)}$ according to the decryption results of $V_{\pi_i(j)}$. CP uses π_i^{-1} to get A_j and B_j. By running *SM* protocol, $E_{PK_c}(H_j)$ and P_{ID_j} can be updated by B_j and A_j, respectively. By using (10), the maximum probability of disease name can be added to $\langle P_{ID_j}\rangle_{j=1,\ldots,n_d}$. Moreover, by using (11) the plaintext of maximum diagnosed probability can be clear to 0, then go on to find next-maximum probability tuple. After finishing *TOP-K*, PA can get encrypted disease names $\langle P_{ID_j}\rangle_{j=1,\ldots,n_d}$. Using private key SK_c, PA can only get top-k probability of disease names. Other disease names are cleaned to 0.

In order to achieve *Step 1* in *Phase C*, CP first needs to judge whether PA suffers from some specific diseases t by running the *PMAX* as follows: $T_t = PMAX(\hat{T}_{t,0}, \hat{T}_{t,1})$ $(t = 1, \ldots, n_d)$. After getting $T_t (t = 1, \ldots, n_d)$, *TOP-K* should find top-k disease names, the process is implemented in *Step 2* of *Phase C*. Notice that there are only k real disease names in ID' and other $n_d - k$ disease names are 0. All the calculations executed by CP can also be run by PU. This is because that the relationship among the ciphertexts will not be exposed by running *TOP-K*.

Algorithm 2: PRIVACY-PRESERVING TOP-k DISEASE NAMES RETRIEVAL PROTOCOL (TOP-K)

```
Input: CP has $n_d$ ciphertext $T_1, \cdots, T_{n_d}$, $(k < n_d)$ , PA holds
       private key $SK_c$.
Output: CP can get top-$k$ disease names.
Initialize set $S'_a$ as $S'_a = \{T_1, \cdots, T_{n_d}\}$ and calculate
$P_{ID_j} = E_{PK_c}(0)$.
for $i = 1$ to $k$ do
    run $T_{MAX}$ =$PMAX_n(T_1, \cdots, T_{n_d})$ to get tuple $T_{MAX}$
    with maximum probability, where $T_1, \cdots, T_{n_d} \in S'_a$.
    for $j = 1$ to $n_d$ do
        randomly choose $R_j \in \mathbb{Z}_N$, calculate:
            $V_j = (E_{PK_c}(H_{MAX}) \cdot E_{PK_c}(H_j)^{N-1})^{R_j}$.   (9)
    permute $n_d$ encrypted data using $\pi_i$ denote as $V_{\pi_i(j)}$, send
    $V_{\pi_i(j)}$ to PA.
    (@PA): Decrypt $V_{\pi_i(j)}$ and by using $SK_c$ and denote as
    $\beta_j = D_{SK_c}(V_{\pi_i(j)})$.
    if $\beta_j = 0$ then
        denote $A'_{\pi_i(j)} = E_{PK_c}(0)$ and $B'_{\pi_i(j)} = E_{PK_c}(1)$.
    else
        denote $A'_{\pi_i(j)} = E_{PK_c}(1)$ and $B'_{\pi_i(j)} = E_{PK_c}(0)$.
    Send $A'_{\pi_i(j)}$, $B'_{\pi_i(j)}$ back to CP.
    (@CP) get $A'_j, B'_j$ by using permutation $\pi_i^{-1}$.
    Refresh $P_{ID_j}$ and $E_{PK_c}(H_j) \in S'_a$ by using SM protocol:
            $P_{ID_j} = P_{ID_j} \cdot SM(E_{PK_c}(ID_j), B'_j)$.   (10)
            $E_{PK_c}(H_j) = SM(E_{PK_c}(H_j), A'_j)$.   (11)
Return $\langle P_{ID_j}\rangle_{j=1,\cdots,n_d}$ to PA.
```

The Necessity of TOP-K in Phase C

The trivial solution of *Phase C* is quite simple; directly send all these encrypted diagnosis results back to the PA. The PA simply decrypts these vectors and gets $\langle \hat{H}_{a,b}, \hat{ID}_{a,b}\rangle$. Then, the patient can get n_d tuples $\langle H_a, ID_a\rangle$ by comparing $H_{a,0}$ and $H_{a,1}$, where $H_a = \max(\hat{H}_{a,0}, \hat{H}_{a,1})$ and $ID_a = \hat{ID}_{H_a}$. Next, the patient will rank all the probability H_a by himself to get top-k probability values and its corresponding disease names. This idea is simple, however, it may leak PU's privacy. For example, if patient only has one symptom (for simplicity, we suppose $X_1 = 1$), the patient encrypts this symptom vector as $E_{PK_c}(X) = (E_{PK_c}(1), E_{PK_c}(0), \ldots, E_{PK_c}(0))$ and sends it to PU for diagnosis. PU can use the encrypted symptom vector to compute $E_{PK_c}(P(A_1 = X_1|C_t = 1)P(C_t = 1))$ and $E_{PK_c}(P(A_1 = X_1|C_t = 0)P(C_t = 0))$. After diagnosis, the patient's encrypted diagnosis vector can be permuted and sent back to CP. Once the encrypted diagnosis vectors are received, patient can use the private key SK_c to decrypt these encrypted values as $\langle P(A_1 = X_1|C_t = 1)P(C_t = 1), ID_t\rangle$ and $\langle P(A_1 = X_1|C_t = 1)P(C_t = 0), ID_t\rangle$ $(k = 1, \ldots, n_d)$. Due to $P(C_t = 1) = 1 - P(C_t = 0)$, the patient can easily know all the $P(A_j = X_1|C_t = 1)$ and $P(A_j = X_1|C_t = 0)$. It cannot satisfy privacy requirements defined in Section III-B. In practical scenarios, patient only cares about his diagnosis results rather than diagnosed probability $H_{t,1}$ and $H_{t,0}$. In this case, CP only needs to send the encrypted top-k diagnosed disease names back to PA. The key idea in *Phase C* of our proposed PPCD is to retrieve encrypted top-k diagnosis results from the universal encrypted diagnosis result set without leaking probability $\hat{H}_{a,1}$ and $\hat{H}_{a,0}$ to PA.

V. Privacy Analysis

In this section, we analyze our proposed PPCD to check whether it can achieve the privacy requirements illustrated in Section III-B.

A. Privacy of Phase A

Because the *Phase A* of the PPCD is depended on *AHPA* scheme, we first analyze the privacy of the *AHPA*, and then show our *Phase A* of PPCD can protect PD's privacy simultaneously.

1) Privacy of AHPA: The *AHPA* scheme is based on traditional El-Gamal encryption scheme [17]. Here, we analyze that our *AHPA* can protect PD's privacy, even CP and PU collude. By using the *Encrypt* algorithm of *AHPA*, $x^{(i)}$ can be encrypted. The original ciphertext are formed as $\{g^{r_i}, e(g,g)^{x^{(i)}} e(g,g)^{a_{i,1} r_i}\}$. The security of the original ciphertext can be reduced to the following hard problem: given $\{g^a, e(g,g)^r\}$, it is still hard to distinguish whether $e(g,g)^{ar}$ is equal to $e(g,g)^z$ or not, where $z \in \mathbb{Z}_p$. This assumption is harder than DBDH assumption (given g^a, g^b, g^c, it is hard to distinguish whether $e(g,g)^{abc}$ is equal to $e(g,g)^z$). This is because give $e(g,g)^{ab}$, it is still hard to get g^a and g^b. If CP gets the original ciphertext, even CP and PU collude, CP or PU cannot get $x^{(i)}$ without knowing $a_{i,1}$ ($a_{i,1}$ is DP i's private key which is kept privately by DP i).

Once the original ciphertext is sent to CP, CP uses $sk_{a_i \to P}$ to convert ciphertext in i's domain into PU's domain. However, PU cannot decrypt the individual converted ciphertext $[x^{(i)}]_{pk_p}$ only using his own private key, or even with re-encryption key $sk_{a_i \to P}$. It is because $e(g,g)^{r'_i r_i}$ are introduced to $[x^{(i)}]_{pk_p}$ during the re-encryption. In order to get the plaintext, PU needs to get $g^{r'_i}$ in order to compute $e(g,g)^{r'_i r_i}$. However, PU cannot get $g^{r'_i}$ only by giving DP i's public key $e(g,g)^{a_{i,1}}$, the re-encryption key $g^{a_{a,1} p_2} g^{r'_i}$, and PU's private key p_2 (given $e(g,g)^{a_{i,1}}$, it is hard to compute $g^{a_{i,1}}$). When encrypted message from all the DP i are aggregated, PU can successfully get the aggregated message by using R and his own private key p_2.

2) Privacy of Phase A: In order to train naïve Bayesian classifier, all the historical medical data should be sent to PU for processing. However, these data cannot be sent to PU directly because of the external adversary. In order to protect historical medical data from eavesdropping during the transmission, all the individual historical medical data should be encrypted by using *AHPA* scheme with DP's public key. Because only DP holds his own private key, other parties cannot decrypt the ciphertext without DP's public key. Once the encrypted historical medical data are received, CP uses the reencryption key to transfer the ciphertext in DP i's domain into PU's domain. Then, CP aggregates these historical medical data into one encrypted vectors. It is because the individual data contain some sensitive information which is related to patient's privacy. Even if the CP and PU are colluded, i.e, CP directly sends encrypted historical medical data or transferred encrypted historical medical data to PU without aggregation, PU still cannot get the DP i's medical data due to the privacy of *AHPA* scheme (see Section V-A2 for detailed privacy analysis). Once received the aggregated encrypted vector, PU can decrypt it and train the naïve Bayesian classifier. Because of the aggregation technique, PU cannot know individual DP i's historical medical data by only using aggregated medical data. Therefore, the individual historical medical data are privacy preserving and *Phase A* satisfies the privacy requirements.

B. Privacy of Phase B

In order to compute disease risk of the patient, PA's symptom information should be sent to PU for prediction. To prevent external adversary from eavesdropping, all the symptom information should be transmitted as encrypted form. External adversary cannot decrypt the ciphertext without PA's private key. In addition, PU cannot get the plaintext of the PA's symptom information. In order to predict PA's disease risk by using the naïve Bayesian classifier, some calculations must be done over the ciphertext. Thanks to additive homomorphism property of Paillier encryption [10], $E_{PK_c}(1 - X_j)$ can be easily calculated. The $E_{PK_c}(X_j)$. $E_{PK_c}(P(A_j = X_j | C_t = 1))$ can be calculated by using (1) without decryption. It is because $P(A_j = 1 | C_t = 1)$ and $P(A_j = 0 | C_t = 1)$ are stored as plaintext form in PU and the scalar-multiplicative homomorphism of Paillier encryption can be used to compute $E_{PK_c}(X_j)$. $E_{PK_c}(P(A_j = X_j | C_t = 1))$. Furthermore, $\hat{K}_{t,1}$ and $\hat{K}_{t,0}$ are needed to be calculated. From (2), we can see that multiplication of the plaintext is needed. Fortunately, *SM* protocol should be used in this phase and its privacy has been proved in [11]. Nothing will be leaked to both sides during the execution. Because all the calculations in *Phase B* are over ciphertext, PU cannot get any information from this procedure, privacy-preserving computing can be achieved in *Phase B*.

C. Privacy of Phase C

In order to prove the privacy of *Phase C* in PPCD, the privacy of *PMAX*, *PMAX*$_n$, and *TOP-K* should be proved first. We will use simulation model defined in secure two-party protocols for semihonest adversaries [22], [23] to prove these three protocols. This model is widely used to prove the privacy of multiparty protocols. We will illustrate this model as follows.

We say a protocol is privacy preserving if each party in the protocol can be computed based on its own inputs and outputs only. We require that a party's view in the protocol execution can be simulated only given its own input and output. It is indicated that each party learns nothing from the protocol by executing itself. Let View_B^Π be the real view of party B when interacting with party A with private input x. Party A's privacy can be guaranteed if there exists a simulator Sim_B such that for any x, $\mathrm{Sim}_B(y, f_1(x,y))$ can generate a view indistinguishable from the B's view in the execution of the real protocol, that is

$$\{\mathrm{Sim}_B(y, f_1(x,y))\}_{x,y\in\{0,1\}^*} \overset{c}{\equiv} \{\mathrm{View}_B^\Pi(x,y)\}_{x,y\in\{0,1\}^*}.$$

The B's privacy means that party B cannot get extra information except those derived from x and $b = f(x,y)$. Its privacy can be achieved if there exists a simulator Sim_A such that for any y with $f_2(x,y) = b$, Sim_A can generate a view indistinguishable from the view of party B in the real execution, that is

$$\{\mathrm{Sim}_A(x, f_2(x,y))\}_{x,y\in\{0,1\}^*} \overset{c}{\equiv} \{\mathrm{View}_A^\Pi(x,y)\}_{x,y\in\{0,1\}^*}.$$

1) Privacy of PMAX: We first illustrate that our *PMAX* can achieve PU's privacy. We construct a simulator Sim_A, which simulates the protocol by selecting two randomly encrypted tuples $\langle X_1, Y_1\rangle$ and $\langle X_2, Y_2\rangle$ as input, and letting y as PA's input.

We now illustrate that the view generated by Sim_A is indistinguishable from the real view in protocol. Sim_A first randomly selects s. By using $\langle X_1, Y_1\rangle$ and $\langle X_2, Y_2\rangle$, if $s = 1$, Sim_A can calculate C_1', C_2', C_3' by using (3)–(5), respectively. If $s = 0$, Sim_A can calculate C_1', C_2', C_3' by using (6)–(8), respectively.

The view generated by Sim_A is $\{y, C_1', C_2', C_3'\}$ and the view in the real execution is $\{y, C_1, C_2, C_3\}$. Because of the privacy of Paillier cryptosystem C_1 and C_1' are indistinguishable. The same is true for C_2 and C_2', C_3 and C_3'. Thus, the sequence of the ciphertexts generated by Sim_A are computationally indistinguishable from the sequence in the real execution for fixed output s. Once C_1', C_2', C_3' are received, PA can calculate D_2', D_3' according to the decryption result of C_1'. This simulation procedure is the same as that of the real protocol. Sim_A and the real view View_A^{Π} are indistinguishable.

Next, we analyze our *PMAX,* which can achieve PA's privacy. Now, we construct a simulator Sim_U without the private input of PA. We need the view generated by Sim_U to be indistinguishable from the view in the real execution. Sim_U simulates as follows. The input of Sim_U are the comparison results $b \in \{0, 1\}$ and PU's private encrypted tuple $\langle X_1, Y_1\rangle$ and $\langle X_2, Y_2\rangle$. Sim_U uses $\langle X_1, Y_1\rangle$ and $\langle X_2, Y_2\rangle$ to construct the C_1, C_2, C_3 for the first step. For the second step, Sim_U will generate the encryption called M according to the value of b. If $b = 0$, the Sim_U generates $M = E_{PK_c}(0)$. If $b = 1$, the Sim_U generates a random encryption as M. Finally, the view generated by Sim_U is $\{\langle X_1, Y_1\rangle, \langle X_2, Y_2\rangle, C_1, C_2, C_3; M, b\}$.

Because C_1, C_2, C_3 are constructed by using the value $\langle X_1, Y_1\rangle$ and $\langle X_2, Y_2\rangle$, the distribution is identical to that of the real execution. For fixed output b, the sequence of the ciphertexts are computationally indistinguishable from the sequence in the real execution. Thus, PU cannot get PA's private input.

PMAX $_n$ just calls *PMAX* as subprotocol and all the messages are encrypted by using Paillier encryption. Due to the privacy of *PMAX*, no information are leaked to PU and PA.

2) Privacy of TOP-K: The protocol will run for *k* times and the simulation procedure is similar. For simplicity, we only take one round for example to illustrate.

We first illustrate our *TOP-K,* which can achieve PU's privacy. We construct a simulator Sim_A, which simulates the protocol by randomly selecting the encryption tuples $T_1', \ldots, T_n'$ as input. Using these random encryptions, PU can use *PMAX*$_n$ to calculate T_{MAX} and then calculate V_j' by. Due to the privacy of the Paillier encryption and *PMAX*$_n$, all V_j' are indistinguishable from V_j in real execution. Thus, this simulation is the same as that of the real protocol. Sim_A and the real view View_A^{Π} are indistinguishable.

Next, we analyze our *TOP-K*, which can achieve PA's privacy. Now, we construct a simulator Sim_U without private input of PA. We need the view generated by Sim_U to be indistinguishable from the view of CP in the real execution. Sim_U simulates as follows. The inputs of Sim_U are the comparison result $b \in \{0, 1\}$ and PU's private input $T_1, \ldots, T_n$. Sim_U uses $T_1, \ldots, T_n$ to construct $V_1, \ldots, V_n$ by using (9) for the first step. For the second step, Sim_U will generate the sequence $c_1, \ldots, c_n$ and $d_1, \ldots, d_n$ according to the result value b. If $b = 0$, the Sim_U generates one encryptions with plaintext 0 from $c_1, \ldots, c_n$ and $n - 1$ other encryptions of 1. Also, the Sim_U generates one encryption with plaintext 0 from $d_1, \ldots, d_n$ and $n - 1$ others encryptions of 0. If $b = 1$, the Sim_U generates n encryption of 1 and denotes them as $c_1, \ldots, c_n$. Also, Sim_U generates n encryption of 0 and denotes them as $d_1, \ldots, d_n$. Finally, the view generated by Sim_U is $\{T_1, \ldots, T_n, V_1, \ldots, V_n; c_1, \ldots, c_n, d_1, \ldots, d_n, b\}$. Since $c_1, \ldots, c_n$ and $d_1, \ldots, d_n$ are constructed by using the value $T_1, \ldots, T_n$, the distribution is identical to that in the real execution. For fixed output b, the sequence of the ciphertexts are computationally indistinguishable from the sequence in the real execution. Thus, CP cannot get PA's private input. Moreover, CP needs to use *SM* protocol to calculate $E_{pk_A}(b_i')$ and $E_{pk_A}(y_i')$ by using (10) and (11) respectively. Due to the privacy of *SM* protocol (see [11] for detail), both the PU's privacy and PA's privacy can be achieved by running this protocol.

3) Privacy of Phase C: The privacy of *Step 1* in *Phase C* can be achieved by running *PMAX*. The privacy of *TOP-K* can be achieved since all the calculations are manipulated over ciphertext, the privacy of *PMAX*$_n$, and the privacy of *SM* protocol. By running *TOP-K*, CP will send n_d encryptions of disease name (include top-k disease names, others are encryption of 0 to PA). Due to the privacy of *TOP-K*, CP does not know the relationship among the ciphertexts, which keeps the privacy of PA in *Step 2*. Finally, only the name of disease will be sent to PA, which guarantees the privacy of PU. Thus, the privacy requirements can be satisfied in *Phase C,* which achieves privacy-preserving computation.

D. Resist to Collusion Attack

In this section, we analyze that our PPCD can achieve the privacy requirements, which resist collusion attack between CP and PU. In *Phase A*, DP i uses pk_{a_i} to encrypt the message and sends $[x^{(i)}]_{pk_{a_i}}$ to CP. Due to the character of *AHPA* scheme, even though CP and PU collude, $x^{(i)}$ cannot be decrypted from the original ciphertext $[x^{(i)}]_{pk_{a_i}}$ or the reencrypted ciphertext $[x^{(i)}]_{pk_p}$ (see Section V-A2 for detailed analysis). In *Phase B*, PA's symptoms are encrypted by using PK_c. Due to the privacy of Paillier encryption scheme, even CP and PU collude, neither CP nor PU can get PA's symptoms without knowing PA's private key. In *Phase C*, due to the privacy of *TOP-K*, CP does not know the relationship among the ciphertext (generate n_d new ciphertext from original n_d ciphertext, include k disease names, others are encryption of 0). Even though CP and PU collude, PU cannot know the patient's top-k diseases without PA's private key SK_c. It can be guaranteed by the security of the Pailliar encryption scheme [10] (adversary cannot get plaintext from ciphertext without knowing the private key).

VI. Performance Analysis

In this section, we analyze the performance of the proposed PPCD in terms of computation cost and communication overhead.

TABLE II
EFFECTIVENESS OF PPCD OVER THE REAL DATASET AID

Disease name	IUB	NRPO
YES	59/59(100%)	50/50(100%)
NO	45/61(73.77%)	60/70(85.71%)
Overall	104/120(86.67%)	110/120(91.67%)

A. Computation Cost

We evaluate computation cost of PPCD by using a custom simulator built in Java. This experiment was run on a test machine with one 2.5-GHz two-core processor and 6-GB RAM. In the experiment, we consider two datasets. One real dataset is used from the UCI machine learning repository called acute inflammations dataset (AID) [24]. We use this dataset to test the performance of the naïve Bayesian classifier by using our PPCD. We also use synthetic dataset to test all factors which affect the performance of PPCD.

1) Real Dataset (AID): The AID was created by a medical expert as a dataset to test the expert system, which was used to perform the presumptive diagnosis of two diseases of the urinary system. This dataset contains 120 instances. Each instance contains six attributes [temperature; occurrence of nausea; lumbar pain; urine pushing; micturition pains; burning of urethra, itch, swelling of urethra outlet] and two decisions [inflammation of urinary bladder (IUB); nephritis of renal pelvis origin (NRPO)]. All the attribute and decisions can be expressed as binary bit 1 (YES) or 0 (No) except for temperature. The value of temperature is varied from 35.5 to $41.5\,^{\circ}$C in the dataset. Before we use this dataset to classify, all the records should be normalized to 56 attributes (including 51 attributes represent for temperature) and two decisions. We first use PPCD to train naïve Bayesian classifier and, then, use this classifier and AID to test the success rate of the classifier. The result is shown in Table II. From the table, we can see that all the instances, which have IUB and NRPO can be successfully classified without false negative diagnosis. However, there exists false positive diagnosis in AID. The false positive rate of IUB and NRPO diagnosis are 26.23% and 14.29%, respectively. We also test the running efficiency about PPCD. In *Phase A*, it takes 8.848 s for PU to train the classifier (including 1.368 s for DP i to encrypt all the symptoms and diseases offline). It takes 187.244 s for CP and PA to compute a new PA's disease risk in *Phase B* (including 2.490 s for DP i to encrypt all the symptoms offline). In *phase C*, it takes 8.724 s for CP and PA to retrieve top-1 (most related diagnose result) by running *TOP-K* (include 2.490 s for DP i to initial *TOP-K* offline).

2) Synthetic Dataset: In order to test all the factors that affect our PPCD, we use the synthetic dataset to test. The randomly generated synthetic dataset consists of 500 tuples with 70 attributes. The value of each elements is randomly picked from 0 to 1. There are four factors which affect the total running time of PPCD: the number of historical medical data (NHMD), the number of symptom attributes (NSA) contained in data, the number of diseases (ND) needed to be classified, and the number of diagnosis disease results (NDDR) needed to be retrieved. In Fig. 4(a)–(c), we plot the running time (including total running time and offline running time) of the PPCD vary with NHMD. Only the running time of *Phase A* increases with the number of data, where other phases are not affected. This is because *Phase A* needs to aggregate more data as NHMD increases. It requires more computational resources, while other phases will not be affected. In Fig. 4, we plot the total running time of PPCD which varies with NSA in dataset, from the figure we can see that the running time of *Phase A* and *Phase B* increases with the number of attributes. With the number of attributes increase, CP needs more multiplications in order to aggregate the ciphertext in *Phase A* and calls more *SM* protocol to calculate $E_{PK_c}(\prod_{j=1}^{n_s} P(F_j = X_j | C_t = 1) \cdot P(C_t = 1))$ in *Phase B*. In Fig. 4, we plot the offline running time of PPCD, which varies with NSA in dataset. The running time of all these phases increases with NSA because more tuples are needed to be encrypted. In Fig. 4, we plot the total running time of PPCD varies with NDDR. From the figure, we can see that *Phase A* and *Phase B* will not vary with k, which means it will not introduce any computations to *Phase A* and *Phase B*. More loops are required, which introduces more computation cost to *Phase C*. In Fig. 4, we plot the offline running time of PPCD varies with NDDR. The running time of all these phases are not affected because no extra tuples is needed to be encrypted during the initialization. In Fig. 4, we plot the total running time of the PPCD vary with ND. It can be seen that more multiplications are required in *Phase A* and more *SM* protocol are called in *Phase B* with increasing of ND. More comparisons are also needed in order to retrieve top-k disease names in *Phase C*. In Fig. 4, we plot the offline running time of PPCD varies with ND. The running time of all these phases are increased because more tuples are needed to be encrypted during the initialization.

B. Communication Overhead

In the following, we discuss the communication cost in our PPCD. The privacy parameter of Paillier encryption system used is 2048 bits. In order to train naïve Bayesian classifier, all historical medical data should be encrypted and sent to CP which costs $\mathcal{O}(l \cdot (n_s + n_d))$ to transmit. Then, CP needs to aggregate all the historical medical data into one vector. This aggregated vector costs $\mathcal{O}(n_s + n_d)$ to transmit to PU. So the total communication cost of *Phase A* is $\mathcal{O}(l \cdot (n_s + n_d))$. In our *AHPA* scheme, one ciphtertext tuple needs 2048 bits to store, and it costs $(l+1) \cdot (n_s + n_d) \cdot 2048$-bits for CP to store all the data in *Phase A*. Extra $(l+1) \cdot (n_s + n_d) \cdot 2047$ bits is needed for storage in CP due to adoption of a privacy-preserving technique (ciphertext expansion). The nonprivacy-preserving technique's communication overhead of *Phase A* is $\mathcal{O}((l+1)(n_s + n_d))$. In order to calculate PA's diseases, its encrypted symptom vector should be sent to PU to process, which costs $\mathcal{O}(n_s)$ to transmit from PA to PU. After computation, the encrypted diagnosed probability and encrypted disease names should be sent to CP to store which costs $\mathcal{O}(n_d)$. Therefore, the total communication cost of *Phase B* is $\mathcal{O}(n_s + n_d)$ and it also costs CP $(n_s + 2n_d) \cdot 2048$ bits to store PA's ciphertext (include n_d

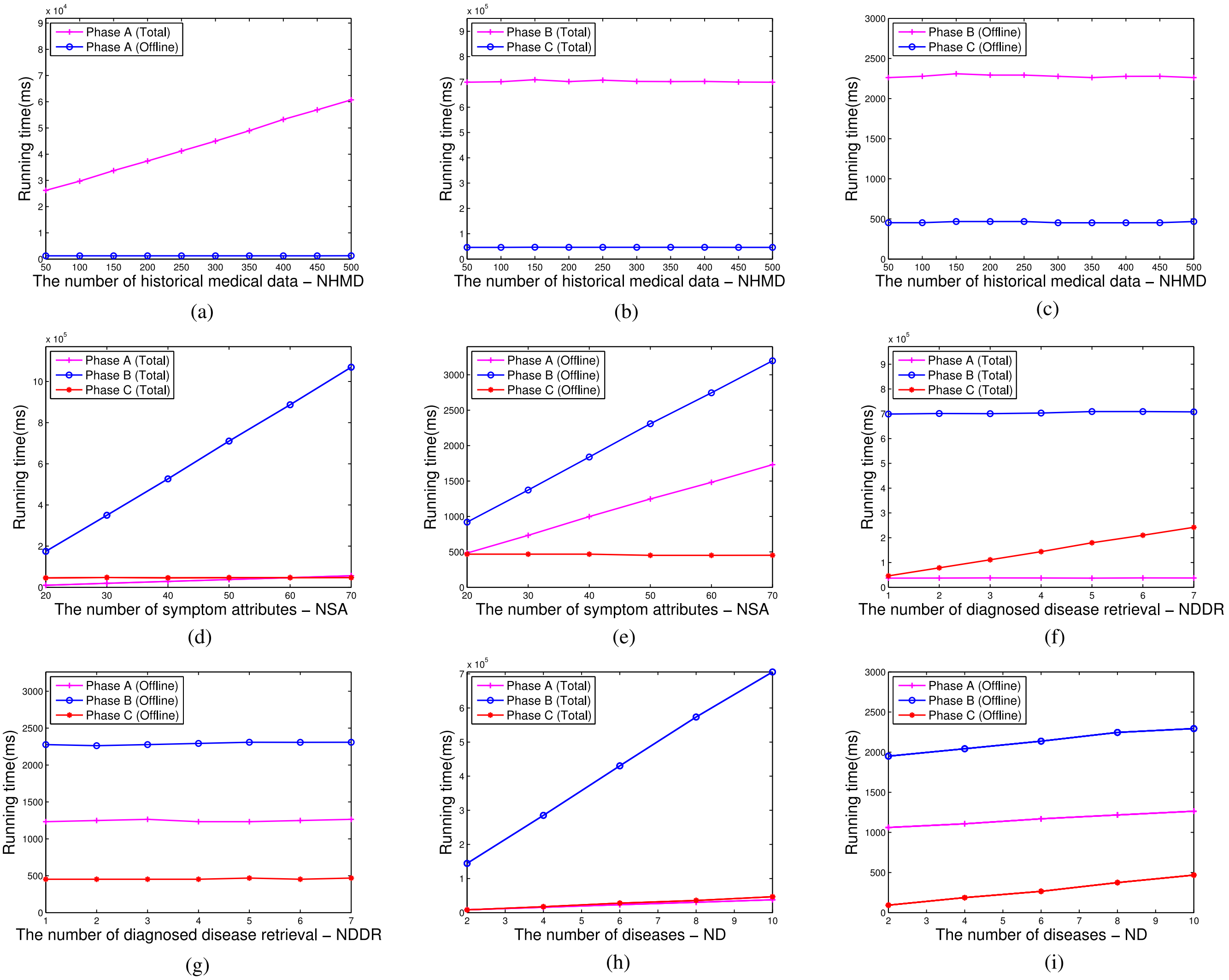


Fig. 4. Computation cost over synthetic dataset. (a) NSA = 40, ND = 10, NDDR = 1. (b) NSA = 40, ND = 10, NDDR = 1. (c) NSA = 40, ND = 10, NDDR = 1. (d) NHMD = 200, ND = 10, NDDR = 1. (e) NHMD = 200, ND = 10, NDDR = 1. (f) NHMD = 200, ND = 10, NSA = 40. (g) NHMD = 40, ND = 10, NSA = 40. (h) NHMD = 200, NSA = 0, NDDR = 1. (i) NHMD = 200, NSA = 10, NDDR = 1.

encrypted symptoms, n_d encrypted disease risks, and n_d encrypted disease names). The total communication cost of nonprivacy-preserving technique of *Phase B* is $\mathcal{O}(n_s + n_d)$ and it also costs CP extra $(n_s + 2n_d) \cdot 2047$ bits to store due to the usage of encryption technique. In *Phase C*, it first costs $\mathcal{O}(n_d)$ (1 round, between CP and PA by call *PMAX*) to achieve *Step 1*, then needs $\mathcal{O}((n_d)^2)$ to calculate $PMAX_n$ ($\log_2 n_d$ rounds). After that, it takes $\mathcal{O}(k \cdot (n_d)^2)$ ($k(\log_2 n_d + 1)$ rounds) to achieve *TOP-K* in *Step 2*. Therefore, the total communication overhead is $\mathcal{O}(k \cdot (n_d)^2)$ ($k \cdot \log_2 n_d + k + 1$ rounds) in *Phase C* for PPCD. It also costs $2n_d \cdot 2048$ bits for CP to store the newly generated ciphertext (include n_d encrypted top-k disease risks, and n_d encrypted top-k disease names). If a nonprivacy-preserving technique is used, it does not need *Phase C* for selection, which means the nonprivacy-preserving technique does not need extra $\mathcal{O}(k \cdot (n_d)^2)$ ($k \cdot log_2 n_d + k + 1$ rounds) communication overhead and extra $2n_d \cdot 2048$ bits for storage in CP.

VII. Related Work

The computer-assisted CDSS was proposed by Ledley and Lusted [25] who found that physicians have an imperfect knowledge of how they solve diagnostic problems. This paper dealt with Bayesian and decision-analytic diagnostic systems and experimental prototypes appeared within a few years [3]. Warner *et al.* [26] developed the first operational Bayesian CDSS for the diagnosis of congenital heart diseases based on history, physical exam, and cardiac catheterization findings. Schurink *et al.* [5] discussed computer-based decision-support systems to assist intensive care unit physicians in the management of infectious diseases. In this paper, they described several computer models (such as bayesian networks) that may be used in clinical practice in the near future. As the privacy of the patient's information becomes more and more important, naïve Bayesian classification were considered as a challenge to privacy preservation due to their natural tendency to use sensitive information about individuals. Privacy-preserving naïve Bayesian classifier was first proposed in [28], where data are horizontal partitioned (different patient's tuples with same attribute are partitioned) and stored distributively in different sites. Kantarcıolu *et al.* [28] achieved naïve Bayesian classifier by using the secure sum protocol [29], which can support both nominal attributes and numeric attributes. Later, Yi and Zhang [30] improved the [28] by both efficiency and privacy and this scheme could prevent eavesdropping attack. This two-party protocol can also be easily extended to multiparty protocol. Different from horizontal partition, another kind of data partition called vertically partition (one patient's different attributes are partitioned) were introduced to privacy-preserving naïve Bayesian classifier by using secure scalar product protocol [31], [32]. Vaidya *et al.* [33] gave us a comprehensive study on both vertically as well as horizontally partitioned data. The data in the existing privacy-preserving naïve Bayesian classifier scheme were distributively stored in different parties as a part of the whole data space. One party should manage and store these data as plaintext. With the development of cloud computing technique, outsourcing the encrypted data to cloud server to store was more common [34], [35]. However, cloud server was always a third-party servers. Storing the patient health data in the third-party servers entailed serious threats to data privacy. Therefore, it was imperative for user to store and manage the outsourced data in a privacy-preserving way. Li *et al.* [36] gave a novel patient-centric framework and a suite of mechanisms for data access control to personal health record stored in cloud servers. It provided a patient-centric model of scalable and secure health information exchange. Elmehdwi *et al.* [8] proposed a scheme which allows user to make k-nearest neighbor (k-NN) query on oursourced encrypted database. This scheme also gave lots of secure primitives and they used these primitives to construct k-NN classifier in [11]. Ayday *et al.* [37] used logistic regression model to compute disease risk privately by using genomic, clinical, and environmental data. Recently, Rahulamathavan *et al.* [38] proposed a privacy-preserving clinical decision support system using a Gaussian kernel-based support vector machine (SVM). During the diagnosis process, patients' data always remained in the encrypted form which can protect patients's privacy. More comprehensive study about privacy-preserving SVM classification can be found in [39]–[41].

VIII. Conclusion

In this paper, we have proposed a PPCD using naïve Bayesian classifier. By taking the advantage of emerging cloud computing technique, PC can use big medical dataset stored in CP to train naïve Bayesian classifier, and then apply the classifier for disease diagnosis without compromising the privacy of DP. In addition, the patient can securely retrieve the top-k diagnosis results according to their own preference in our system. Since all the data are processed in the encrypted form, our system can achieve patient-centric diagnose result retrieval in privacy-preserving way. For the future work, we will exploit PPCD with other advanced data mining techniques, such as SVM classification.

References

[1] (2013). Transforming health care through big data strategies for leveraging big data in the health care industry. [Online]. Available: http://ihealthtran.com/wordpress/2013/03/iht%C2%B2-releases-big-data-research-report-download-today/

[2] E. S. Berner, *Clinical Decision Support Systems*. New York, NY, USA: Springer, 2007.

[3] M. A. Musen, B. Middleton, and R. A. Greenes, "Clinical decision-support systems," in *Biomedical Informatics*. New York, NY, USA: Springer, 2014, pp. 643–674.

[4] H. Monkaresi, R. A. Calvo, and H. Yan, "A machine learning approach to improve contactless heart rate monitoring using a webcam," *IEEE J. Biomed. Health Informat.*, vol. 18, no. 4, pp. 1153–1160, Jul. 2014.

[5] C. Schurink, P. Lucas, I. Hoepelman, and M. Bonten, "Computer-assisted decision support for the diagnosis and treatment of infectious diseases in intensive care units," *Lancet Infectious Dis.*, vol. 5, no. 5, pp. 305–312, 2005.

[6] I. Kononenko, "Machine learning for medical diagnosis: History, state of the art and perspective," *Artif. Intell. Med.*, vol. 23, no. 1, pp. 89–109, 2001.

[7] N. Lavrač, I. Kononenko, E. Keravnou, M. Kukar, and B. Zupan, "Intelligent data analysis for medical diagnosis: Using machine learning and temporal abstraction," *Artif. Intell. Commun.*, vol. 11, no. 3, pp. 191–218, 1998.

[8] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest neighbor query over encrypted data in outsourced environments," in *Proc. IEEE 30th Int. Conf. Data Eng.*, pp. 664–675, 2014.

[9] K. M. Leung, "Naive Bayesian classifier," *Tech. Notes Polytechnic Univ. Dept. Comput. Sci./Finance Risk Eng.,* (2007). [Online]. Available: http://cis.poly.edu/ mleung/FRE7851/f07/naiveBayesianClassifier.pdf

[10] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Adv. Cryptol. Int. Conf. Theory Appl. Cryptograp. Techn*., Prague, Czech Republic, May 2–6, 1999, pp. 223–238.

[11] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "k-nearest neighbor classification over semantically secure encrypted relational data," *IEEE Trans. Knowledge Data Eng.*, (2015). [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6930802.

[12] A. McCallum and K. Nigam, "A comparison of event models for naive Bayes text classification," in *Proc. Artif. Intell. Workshop Learn. Text Categorization*, 1998, vol. 752, pp. 41–48.

[13] J. Chen, H. Huang, S. Tian, and Y. Qu, "Feature selection for text classification with naïve Bayes," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 5432–5435, 2009.

[14] I. Rish, "An empirical study of the naive Bayes classifier," in *Proc. Int. Joint Conf. Artif. Intell. Workshop Empirical Methods Artif. Intell.*, vol. 3, no. 22, 2001, pp. 41–46.

[15] R. Bellazzi and B. Zupan, "Predictive data mining in clinical medicine: current issues and guidelines," *Int. J. Med. Informat.*, vol. 77, no. 2, pp. 81–97, 2008.

[16] J. L. Hellerstein, T. Jayram, and I. Rish, *Recognizing End-User Transactions in Performance Management*, IBM TJ Watson Research Center, Yorktown Heights, NY, USA, 2000.

[17] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Advances in Cryptology*. New York, NY, USA: Springer, 1985, pp. 10–18.

[18] B. K. Samanthula, Y. Elmehdwi, G. Howser, and S. Madria, "A secure data sharing and query processing framework via federation of cloud computing," *Inf. Syst.*, vol. 48, pp. 196–212, 2015.

[19] T. T. A. Dinh and A. Datta, "Stream on the sky: Outsourcing access control enforcement for stream data to the cloud, arXiv preprint arXiv:1210.0660, (2012). [Online] Available: http://arxiv.org/abs/1210.0660

[20] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Theory of Cryptography*. New York, NY, USA: Springer, 2005, pp. 325–341.

[21] A. Shiryaev, "Bayes formula," in *Encyclopedia of Mathematics.* New York, NY, USA: Springer, 2011.

[22] Y. Lindell and B. Pinkas, "A proof of security of YAOS protocol for two-party computation," *J. Cryptol.*, vol. 22, no. 2, pp. 161–188, 2009.

[23] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2009, vol. 2.

[24] Acute inflammations data set, UCI machine learning repository. (2009). [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Acute+Inflammations/

[25] R. S. Ledley and L. B. Lusted, "Reasoning foundations of medical diagnosis," *Science*, vol. 130, no. 3366, pp. 9–21, 1959.

[26] H. R. Warner, A. F. Toronto, L. G. Veasey, and R. Stephenson, "A mathematical approach to medical diagnosis: Application to congenital heart disease," *J. Amer. Med. Assoc.*, vol. 177, no. 3, pp. 177–183, 1961.

[27] C. Schurink, P. Lucas, I. Hoepelman, and M. Bonten, "Computer-assisted decision support for the diagnosis and treatment of infectious diseases in intensive care units," *Lancet Infectious Dis.*, vol. 5, no. 5, pp. 305–312, 2005.

[28] M. Kantarcıoglu, J. Vaidya, and C. Clifton, "Privacy preserving naive bayes classifier for horizontally partitioned data," in *Proc. IEEE Int. Conf. Workshop Privacy Preserving Data Min.*, 2003, pp. 3–9.

[29] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," *ACM SIGKDD Explorations Newslett.*, vol. 4, no. 2, pp. 28–34, 2002.

[30] X. Yi and Y. Zhang, "Privacy-preserving naive Bayes classification on distributed data via semi-trusted mixers," *Inf. Syst.*, vol. 34, no. 3, pp. 371–380, 2009.

[31] A. Amirbekyan and V. Estivill-Castro, "A new efficient privacy-preserving scalar product protocol," in *Proc. 6th Australas. Conf. Data Min. Analytics*, 2007, pp. 209–214.

[32] R. Lu, H. Zhu, X. Liu, J. K. Liu, and J. Shao, "Toward efficient and privacy-preserving computing in big data era," *IEEE Netw.*, vol. 28, no. 4, pp. 46–50, Jul./Aug. 2014.

[33] J. Vaidya, M. Kantarcioglu, and C. Clifton, "Privacy-preserving naïve Bayes classification," *Very Large Data Bases J.*, vol. 17, no. 4, pp. 879–898, 2008.

[34] A. Abbas and S. U. Khan, "A review on the state-of-the-art privacy-preserving approaches in the e-health clouds," *IEEE J. Biomed. Health Informat.*, vol. 18, no. 4, pp. 1431–1441, Jan. 2014.

[35] Y. Tong, J. Sun, S. S. M. Chow, and P. Li, "Cloud-assisted mobile-access of health data with privacy and auditability," *IEEE J. Biomed. Health Inform.*, vol. 18, no. 2, pp. 419–429, Mar. 2014.

[36] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 131–143, Jan. 2013.

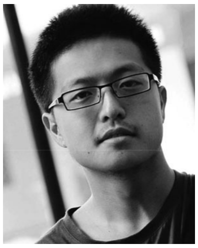
[37] J.-P. Hubaux, J. Fellay, E. Ayday, M. Laren, J. L. Raisaro, P. J. Mclaren, "Privacy-preserving computation of disease risk by using genomic, clinical, and environmental data," in *Proc. USENIX Security Workshop Health Inf. Technol.*, (2013). [Online]. Available: http://dx.doi.org/10.1155/2014/827371.

[38] Y. Rahulamathavan, S. Veluru, R. Phan, J. Chambers, and M. Rajarajan, "Privacy-preserving clinical decision support system using Gaussian kernel based classification," *IEEE J. Biomed. Health Informat.*, vol. 18, no. 1, pp. 56–66, Jan. 2014.

[39] K. Lin and M. Chen, "On the design and analysis of the privacy-preserving SVM classifier," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 11, pp. 1704–1717, Nov. 2011.

[40] H. Yu, X. Jiang, and J. Vaidya, "Privacy-preserving SVM using nonlinear kernels on horizontally partitioned data," in *Proc. ACM Symp. Appl. Comput*. 2006, pp. 603–610.

[41] H. Li, L. Xiong, L. Ohno-Machado, and X. Jiang, "Privacy preserving RBF kernel support vector machine," *BioMed. Res. Int.*, vol. 2014, 2014.



Ximeng Liu (S'13) received the B.Sc. degree in electronic engineering from Xidian University, Xi'an, China, in 2010, where he is currently working toward the Ph.D. degree at the School of Telecommunications Engineering.

He is a Visiting Ph.D. student at the INFINITUS lab, School of Electrical and Electronics Engineering, Nanyang Technological University, Nanyang, Singapore. His research interests include applied cryptography and big data security.

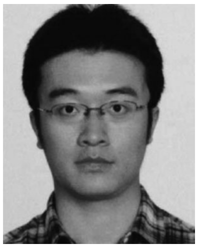
Rongxing Lu (S'09–M'11) received the Ph.D. degree in computer science from Shanghai Jiao Tong University, Shanghai, China, in 2006, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2012.

Since May 2013, he has been with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, as an Assistant Professor. His research interests include computer, network and communication security, applied cryptography, security and privacy analysis for vehicular network, eHealthcare system, and smart grid communications.

Dr. Lu received the Canada Governor General Gold Medal. He also received the IEEE Communications Society Asia Pacific Outstanding Young Researcher Award in 2013.

Jianfeng Ma received the B.Sc. degree in mathematics from Shaanxi Normal University, Xi'an, China, in 1985, and the M.Sc. and Ph.D. degrees in computer software and communications engineering from Xidian University, Xi'an, in 1988 and 1995, respectively.

He is currently a Professor and Ph.D. Supervisor at the School of Computer Science and Technology, Xidian University. His current research interests include information and network security, wireless and mobile computing systems, and computer networks. He has published more than 200 refereed articles in these areas and coauthored more than ten books. He is a Senior Member of the Chinese Institute of Electronics.



Le Chen received the B.Sc. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2007. He is working toward the Ph.D. degree with the same university.

He is currently a Visiting Scholar with the INFINITUS Lab, School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore. His research interests include secure data aggregation, wireless network security, and applied cryptography.

Baodong Qin received the B.Sc. degree in information security and the M.Sc degree in system analysis and integration from Shandong University, Ji'na, China, in 2004 and 2007, respectively. He is currently working toward the Ph.D. degree at the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China.

He is a Research Assistant at the School of Information System, Singapore Management University, Singapore. His research interests include applied cryptography and provable security.