4-2020

Privacy-preserving outsourced support vector machine design for secure drug discovery

Ximeng LIU
Singapore Management University, xmliu@smu.edu.sg

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Kim-Kwang Raymond CHOO
University of Texas at San Antonio

Yang YANG
Fuzhou University

IEEE TRANSACTIONS ON CLOUD COMPUTING, VOL. 8, NO. 2, APRIL-JUNE 2020
DOI: 10.1109/TCC.2018.2799219

Privacy-Preserving Outsourced Support Vector Machine Design for Secure Drug Discovery

Ximeng Liu, *Member, IEEE*, Robert H. Deng, *Fellow, IEEE*,
Kim-Kwang Raymond Choo, *Senior Member, IEEE*, and Yang Yang, *Member, IEEE*

Abstract—In this paper, we propose a framework for privacy-preserving outsourced drug discovery in the cloud, which we refer to as POD. Specifically, POD is designed to allow the cloud to securely use multiple drug formula providers' drug formulas to train Support Vector Machine (SVM) provided by the analytical model provider. In our approach, we design secure computation protocols to allow the cloud server to perform commonly used integer and fraction computations. To securely train the SVM, we design a secure SVM parameter selection protocol to select two SVM parameters and construct a secure sequential minimal optimization protocol to privately refresh both selected SVM parameters. The trained SVM classifier can be used to determine whether a drug chemical compound is active or not in a privacy-preserving way. Lastly, we prove that the proposed POD achieves the goal of SVM training and chemical compound classification without privacy leakage to unauthorized parties, as well as demonstrating its utility and efficiency using three real-world drug datasets.

Index Terms—Cloud-supported drug discovery, privacy-preserving, support vector machine, sequential minimal optimization

1 Introduction

Drug discovery can deliver significant benefits to the society, particularly in an aging society. Drug discovery is generally defined as the process of identifying one or more active ingredients from traditional remedies, and includes the identification of screening hits, medicinal chemistry and optimization of these hits to increase the affinity, selectivity (to reduce the potential of side effects), bioavailability, and metabolic half-life [1]. However, drug discovery is a challenging, costly, and inefficient process with a low rate of discovering new therapeutic uses. For example, drugs can reportedly take 12 years from initial discovery stage to licensing approval, and the Association of the British Pharmaceutical Industry estimated the amount of investment to be at £1.15 billion per drug [2]. In other words, drug discovery requires significant investment from the pharmaceutical sector and governments [3].

- *X. Liu is with the School of Information Systems, Singapore Management University, Singapore 188065, Singapore, and with the College of Mathematics and Computer Science, Fuzhou University, Fujian 350015, China. E-mail: snbnix@gmail.com.*
- *R.H. Deng is with the School of Information Systems, Singapore Management University, Singapore 188065, Singapore. E-mail: robertdeng@smu.edu.sg.*
- *K.-K. R. Choo is with the Department of Information Systems and Cyber Security and Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, TX 78249-0631. E-mail: raymond.choo@fulbrightmail.org.*
- *Y. Yang is with the College of Mathematics and Computer Science, Fuzhou University, Fujian 350015, China. E-mail: yang.yang.research@gmail.com.*

Technologies can play a facilitating role in drug discovery (e.g., in computer-aided drug design to find new biologically active compounds [4]). According to a report from Research and Markets [5], the global drug discovery technologies market is expected to grow at a compound annual growth rate of approximately 12.2 percent over the next decade to reach approximately $160 billion by 2025.

Machine learning is one of the several technologies that can be used in drug discovery. For example, machine learning tools can be used to evaluate the potential biological activity and to provide predictions about the physicochemical and pharmacokinetic properties of chemical structures [6], [7]. Of the data mining tools, Support Vector Machine (SVM) [8] has a relatively high decision rate and has been widely used in recent times to predict ligand-based chemical compounds in drug discovery [9]. In approaches using SVMs, we use existing datasets of known drug formulas to train the SVM classifier, and the trained SVM classifier can be used for new drug compound visual scanning (See Fig. 1). Due to the significant investments and high commercial values involved in drug discovery, privacy is an important factor [10]. For example, how can we minimize the risk of unauthorized disclosure during the SVM training phase? In this context, when a researcher sends some chemical compounds to the cloud for SVM classification, it is important to ensure that the potential new drug compounds will not be leaked to a third-party, such as a competing pharmaceutical corporation.

Furthermore, to train the SVM, multiple pharmaceutical corporations may collaborate in order to increase the SVM decision rate. At the same time, these corporations do not wish to reveal their datasets. How to achieve secure SVM training and decision under multiple data sources without compromising the privacy of each individual party remains

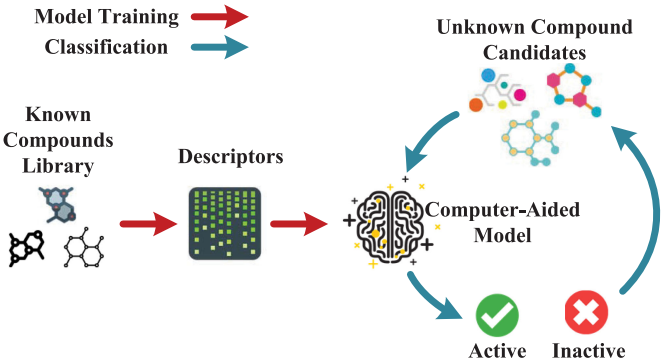


Fig. 1. Drug discovery cycle.

a research and operational challenge. Thus, in this paper, we propose a **P**rivacy-preserving **O**utsourced Support Vector Machine Design for Secure **D**rug discovery in the cloud environment, hereafter referred to as POD. Unlike existing drug discovery frameworks [11], our POD seeks to achieve the following:

- *Secure Outsourced Data Storage:* The drug formula owner can securely outsource the data (e.g., drug formula) to the cloud for storage without leaking the data to unauthorized third parties.
- *Secure Multi-Source SVM Training:* The POD allows an authorized model provider to use other drug formula owners' encrypted data to train the SVM on-the-fly. The model provider can decrypt and obtain the trained model without knowing (contents of) the training dataset.
- *Secure SVM Drug Decision:* An authorized tester can securely upload his/her drug chemical compounds to the cloud and determine whether the compound is active or not in a privacy-preserving way.
- *Mitigating Plaintext Overflow:* During computation, the plaintext length of the ciphertext may increase and exceed the plaintext upper-bound, and therefore, further secure computation will result in the plaintext overflow issue. A secure fast approximation method is then designed to reduce the plaintext size of the ciphertext such that the new ciphertext can be further computed.
- *Ease of Use:* POD does not require the authorized tester to perform any complex pre-processing before outsourcing. Also, the interaction between drug tester and the cloud server is kept to a minimum during secure computation, since the tester only needs to send an encrypted query to the cloud server, and waits for the cloud to reply with the encrypted decision result in a single round.

The remainder of this paper is organized as follows. In Section 2, we present the preliminaries required for the understanding of our proposed POD. In Section 3, we formalize the system model, state the problem, and show the attack model. Then, we describe POD, and the secure computation protocols to solve the overflow issue in POD, in Sections 4 and 5, respectively. The security analysis and performance evaluation are respectively presented in Sections 6 and 7. Related work is discussed in Section 8. Section 9 concludes this paper.

TABLE 1
Summary of Notations

Notation	Definition
pk_a/sk_a	Party a' public key/private key
(ssk_{KGC}, vk_{KGC})	Strong unforgeable signature/verification key
$SK^{(1)}/SK^{(2)}$	Partial Strong private key of DT-PKC scheme
$[x]_{pk_a}/[x]$	Ciphertext of DT-PKC scheme with integer x
$\langle x \rangle$	Encryption of fraction number x
$\|x\|$	Bit-length of x.
CER_x	Certificate for domain x
$K(\vec{x}_i, \vec{x}_j)$	Kernel function of SVM
$\alpha_1^*, \ldots, \alpha_n, \beta$	Parameters of SVM
$\mathcal{S}(\Gamma)$	Size of a set Γ
S^*	Encrypted training dataset with SVM parameters
g, N	Group generator, plaintext domain of DT-PKC

2 Preliminary

In this section, we outline the definition of SVM, which is the basis in our proposed POD. Also, we introduce a basic crypto primitive and the secure computation protocols used as the building blocks for the construction of POD. Table 1 summarizes the key notations used in this paper.

2.1 Support Vector Machine (SVM)

Given n training instances $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$, where $\vec{x}_i$ is t-dimension real vector and $y_i \in \{1, -1\}$. The SVM classifier attempts to find a hyperplane that divides the two classes with the largest margin. The training SVM classifier can be converted into a solver for the SVM dual problem, i.e., $\min_{\vec{\alpha}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i^*\alpha_j^* y_i y_j K(\vec{x}_i, \vec{x}_j) - \sum_{i=1}^{n}\alpha_i^*$, subject to $\sum_{i=1}^{n}\alpha_i^* y_i = 0$ and $0 \le \alpha_i^* \le C$, where $i = 1, \ldots, n$, C is the upper limit of all α_i^*, and $K(\vec{x}_i, \vec{x}_j)$ is the kernel function. Then, we calculate β^* as $\beta^* = y_j - \sum_{i=1}^{n}\alpha_i^* y_i K(\vec{x}_i, \vec{x}_j)$. Using the trained SVM parameters, we can achieve the decision function of SVM as $h(\vec{x}) = Sign(\sum_{i=1}^{n}\alpha_i^* y_i K(\vec{x}_i, \vec{x}) + \beta^*)$, where $Sign(x)$ is the function. The function outputs 1 when $x > 0$, and -1 otherwise.

2.2 Basic Primitive

In the proposed POD, we use the Distributed Two Trapdoors Public-Key Cryptosystem (DT-PKC) [12] as the basic crypto primitive. The latter contains eight algorithms: KeyGen, Encryption (`Enc`), Decryption with weak private key (`WDec`), Decryption with strong private key (`SDec`), Strong private key splitting (`SkeyS`), Partial Decryption Step-1 (`PD1`), Partial Decryption Step-2 (`PD2`), and Ciphertext Refresh (`CR`) (see supplemental material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCC.2018.2799219, for the construction). The overall plaintext domain is in $\mathbb{Z}_N$, and we use plaintext domain $[0, N/2)$ to represent positive number and plaintext domain $(N/2, N-1]$ to represent negative number. As the plaintext domain is modular $N-1$, it has $N - a \equiv -a \bmod N$. The construction can be found in the supplemental material, available online section. We only describe two main homomorphic properties of DT-PKC, as follows: 1. Additive homomorphism: given two ciphertext $[m_1]_{pk}$ and $[m_1]_{pk}$, the additive homomorphism can be achieved by $[m_1]_{pk} \cdot [m_2]_{pk} = [m_1 + m_2]_{pk}$. 2) Scalar-multiplicative Homomorphism: given ciphertexts $[m]$ and a constant number

$c \in \mathbb{Z}_N$, it has $([m]_{pk})^c = [cm]_{pk}$. Specifically, let $c = N-1$ and we have $([m]_{pk})^{N-1} = [-m]_{pk}$. Thus, the secure computation achieves under the same public key pk.

2.3 Secure Integer & Fraction Computation Protocols

All the following protocols are executed between two non-colluding servers, and we refer interested reader to our previous work [13], [14] or the supplemental material, available online section for the concrete construction.

With two encrypted data $[x]$ and $[y]$, Secure Multiplication Protocol (SM) securely outputs $[z]$, s.t. $[z] = [x \cdot y] \leftarrow \texttt{SM}([x],[y])$. The Secure Less Than Protocol (SLT) securely outputs $[u] \leftarrow \texttt{SLT}([x],[y])$, where $u = 0$ if $x \geq y$ and $u = 1$ if $x < y$. The Secure Equivalent Testing Protocol (SEQ) computes $[u] \leftarrow \texttt{SEQ}([x],[y])$, such that $u = 0$ if x is equal to y and $u = 1$ if $x \neq y$. The Secure Bit-Decomposition Protocol (SBD) securely outputs bit-decomposition ciphertexts $([\mathrm{x}_{\mu-1}], \ldots, [\mathrm{x}_0]) \leftarrow \texttt{SBD}([x])$, where $\sum_{i=0}^{\mu-1} \mathrm{x}_i \cdot 2^i = x$, and $\mathrm{x}_i \in \{0,1\}$. The Secure Sign Bit Acquisition Protocol (SSBA) securely outputs ciphertexts $([x^*],[s^*]) \leftarrow \texttt{SSBA}([x])$, s.t., $x^* = x$ and $s^* = 1$ when $x \geq 0$, or $x^* = N - x$ and $s^* = 0$ when $x < 0$. To execute Secure Inner Product Protocol (SIP) with input $[x_0], \ldots, [x_{t-1}]$ and $[y_0], \ldots, [y_{t-1}]$, SIP securely computes inner product, and outputs encrypted integer $[z] \leftarrow \texttt{SIP}([x_0], \ldots, [x_{t-1}]; [y_0], \ldots, [y_{t-1}])$, where $z = \sum_{i=0}^{t-1} x_i \cdot y_i$.

To store the decimal numbers, one decimal number x^* can be stored as two ciphertexts $\langle x^* \rangle = ([x^+],[x^-])$, such that $x^* \approx \frac{x^+}{x^-}$ (we adopt this idea from [13]). Given $([x^+],[x^-])$ and $([y^+],[y^-])$, we construct Secure Fraction Addition (FAdd), Secure Fraction Subtraction (FSub), Secure Fraction Multiplication (FMul), Secure Fraction Division (FDiv), Secure Fraction Comparison (FCmp), Secure Fraction Equality Test (FEqu), and Secure Fraction Inner Product Protocol (FIP) by directly using the secure integer protocols (see [13] or the supplemental material, available online section).

3 System Model & Privacy Requirement

3.1 System Model

The proposed POD system comprises six types of parties, namely: Drug Formula Providers (DPs), Drug Formula Tester (DT), Key Generation Center (KGC), Cloud Platform (CP), Computation Service Provider (CSP), and Analytical Model Provider (AP) – see Fig. 2.

- The *KGC* is trusted by all other entities in the system, and is tasked with distributing and managing all public/private keypairs for the system.
- The *CP* has almost 'unlimited' data storage spaces, and stores and manages data outsourced from all registered parties in the system. It can also perform certain calculations on ciphertexts.
- *CSP* is able to partially decrypt ciphertexts sent by the CP, perform certain calculations, and then re-encrypt the calculated results.
- Each *DP* can be an individual commercial pharmaceutical corporation, which encrypts and forwards its drug formulary to CP for storage. Also, DP can authorize a specific party for outsourced formula processing on-the-fly.

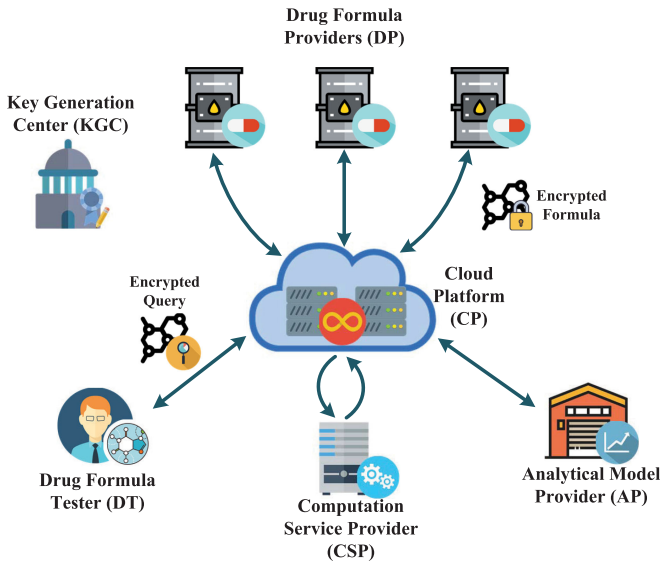


Fig. 2. System model under consideration.

- A *DT* can be a researcher who needs to test some compounds (e.g., determine whether compounds are active for a disease or not). The authorized DT can encrypt these compounds, and send them to the CP for secure classification. Once the encrypted results are received, the authorized DT can decrypt and obtain the classification result.
- An *AP* can be a commercial corporation that provides secure classification model for DT. If the AP is authorized by a DP, then the DP's outsourced formulas can be used for secure model training on-the-fly.

In a real-world deployment, POD can contain multiple DPs and DTs, and each DP or DT can contain multiple drug formulas. However, for ease of understanding, in this paper we assume the POD has n DPs, one DT, one AP, one CSP, one CP, one KGC, and each DP $\mathscr{D}_i$ contains only one drug formula $\vec{x}_i = (x_{i,1}, \ldots, x_{i,t})$ and y_i, where $x_{i,1} \in \mathbb{Z}_p$ and $y_i \in \{1, -1\}$. We define Sig/Verify to be a strong unforgeable signature/verification scheme, and KGC generates the signing and verification key pair (ssk_{KGC}, vk_{KGC}), sends vk_{KGC} to the other parties in the system, and stores ssk_{KGC} locally. The strong private key SK should be randomly split into $SK^{(1)}$ and $SK^{(2)}$ using the SkeyS algorithm, prior sending to CP and CSP for storage, respectively.

3.2 Attack Model

In our attack model, CP, CSP, AP, DPs, and DT are *curious-but-honest* parties that strictly follow the protocol, but are also interested to learn data belonging to other parties. Therefore, we introduce three active adversaries $\mathcal{A}_1^*$, $\mathcal{A}_2^*$, and $\mathcal{A}_3^*$ in our model. The goal of $\mathcal{A}_1^*$, $\mathcal{A}_2^*$, and $\mathcal{A}_3^*$ is to respectively challenge DT's ciphertext, DP's ciphertext, and AP's ciphertext, via the following capabilities: 1) $\mathcal{A}_1^*$, $\mathcal{A}_2^*$, and $\mathcal{A}_3^*$ may *eavesdrop* on all communication links to obtain the encrypted data. 2) $\mathcal{A}_1^*$, $\mathcal{A}_2^*$, and $\mathcal{A}_3^*$ may *compromise* the CP's data storage to get all ciphertexts in the CP, and all ciphertexts sent from the CSP by executing an interactive protocol. 3) $\mathcal{A}_1^*$, $\mathcal{A}_2^*$, and $\mathcal{A}_3^*$ may *compromise* the CSP to guess plaintext values of all ciphertexts sent from the CP by executing an interactive protocol. Note that $\mathcal{A}_1^*$, $\mathcal{A}_2^*$, and $\mathcal{A}_3^*$ are restricted from

compromising both CP and CSP concurrently.[1] Also, $\mathcal{A}_1^*$, $\mathcal{A}_2^*$, and $\mathcal{A}_3^*$ cannot compromise the respective challenger's ciphertext (i.e., DT, DP, and AP). We remark that such restrictions are typical in adversary models used in cryptographic protocols (see the review of adversary models in [12]).

4 Basic Component of POD Framework

Before constructing POD, we will construct the following secure data computation protocols as the basis of POD.

4.1 User Authorization and Key Distribution

4.1.1 Certificate Structure and Revocation

If a party A (e.g., AP) wishes to simultaneous perform some computations on the other parties (e.g., DPs) $D_1, \ldots, D_n$'s ciphertexts, D_i needs to present a valid authorization time for A (e.g., periods of authorization time in the form of $PT_i =$ "$201801012310 - 201801020800$" from D_i), and sends it to KGC. Next, KGC generates a certificate number CN for each certificate and constructs a new certificate $\mathbf{CER}_{\sigma^*}$ with a valid period $PT_{\sigma^*} = PT_1 \cap \cdots \cap PT_n$ and access domain AD[2] as follows: $\langle cer = (CN, AD, \sigma^*, PT_{\sigma^*}, pk_{\sigma^*}); \mathtt{Sig}(cer, ssk_{KGC})\rangle$, where $\sigma^* = \{D_1, \ldots, D_n : A\}$, $pk_{\sigma^*} = g^{sk_{\sigma^*}}$ and $sk_{\sigma^*} = \theta_{\sigma^*}$ are randomly selected from $\mathbb{Z}_N$. To revoke $\mathbf{CER}_{\sigma^*}$ within PT_{σ^*}, KGC generates a revocation certificate $\mathbf{RVK}_{\sigma^*} = \langle rev = (revoke, CN); \mathtt{Sig}(rev, ssk_{KGC})\rangle$. After that, $\mathbf{RVK}_{\sigma^*}$ is sent to CP and CSP to revoke $\mathbf{CER}_{\sigma^*}$.

4.1.2 Certificate Generation and Key Distribution

To distribute keys in the POD, the KGC chooses public parameters $pp = (N, g)$, randomly selects $sk_x = \theta_x \in \mathbb{Z}_N$ (for all $x \in \{\mathscr{D}_1, \ldots, \mathscr{D}_n, \mathscr{A}, \mathscr{B}\}$, where $\mathscr{A}$ denotes the AP, $\mathscr{B}$ denotes the DT, $\mathscr{D}_1, \ldots, \mathscr{D}_n$ denote the DP 1 to DP n, respectively), and computes $pk_i = g^{\theta_i}$ under the same N and g. After that, KGC generates $\mathbf{CER}_{\mathscr{D}_i}$ for $\mathscr{D}_i$ $(i = 1, \ldots, n)$, $\mathbf{CER}_{\mathscr{A}}$ for $\mathscr{A}$, $\mathbf{CER}_{\mathscr{B}}$ for $\mathscr{B}$, where $\mathbf{CER}_x = \langle cer = (CN, x, PT_x, pk_x); \mathtt{Sig}(cer, ssk_{KGC})\rangle$. Also, the private key $sk_{\mathscr{D}_i} = \theta_{\mathscr{D}_i}$ $(i = 1, \ldots, n)$, $sk_{\mathscr{A}} = \theta_{\mathscr{A}}$ and $sk_{\mathscr{B}} = \theta_{\mathscr{B}}$ are privately sent to $\mathscr{D}_i$, $\mathscr{A}$ and $\mathscr{B}$ via secure channel, respectively. Also, all the above certificates are sent to both CP and CSP for storage. Next, $\mathbf{CER}_\sigma$ and $\mathbf{CER}_{\sigma'}$ are generated by KGC, and are only sent to $\mathscr{A}$ and $\mathscr{B}$ for storage respectively, where $\sigma = \{\mathscr{D}_1, \ldots, \mathscr{D}_n : \mathscr{A}\}$ and $\sigma' = \{\mathscr{D}_1, \ldots, \mathscr{D}_n, \mathscr{A} : \mathscr{B}\}$. As these two encrypted domains are only used for secure computation, the verification key vk_{KGC} is sent to both CP and CSP. Moreover, the public keys $pk_\sigma = g^{\theta_\sigma}$ and $pk_{\sigma'} = g^{\theta_{\sigma'}}$ are sent to CP and CSP, and the corresponding private keys $sk_\sigma = \theta_\sigma$ and $sk_{\sigma'} = \theta_{\sigma'}$ are randomly selected from $\mathbb{Z}_N$ and stored only in the KGC. Thus, it is not possible for an external adversary and other internal parties to retrieve and decrypt the ciphertexts in the encrypted domains σ and σ', since these entities do not have access to the corresponding private keys sk_σ and $sk_{\sigma'}$. Specifically, it can be guaranteed by the semantic security of the DT-PKC scheme (see Theorem 1). Next, to support ciphertext calculation under different public keys, we will construct the secure domain transformation protocol.

4.2 Secure Domain Transformation (SDT)

Due to the characteristics of the DT-PKC scheme, homomorphic properties can be achieved only if the ciphertexts are encrypted with the same public key. In reality, ciphertexts computation under different public keys are more practical, and thus, they cannot be directly computed. Liu et al. [15] presented a solution to achieve commonly used integer calculations across multiple keys; however, the computation cost is high. Here, we give another solution called Secure Domain Transformation (SDT) to transform ciphertexts in different keys into a single authorized domain.

SDT is described as follows: Given a ciphertext $[x]_{pk_a}$ with public key pk_a, output $[z]_{pk_\sigma}(z = x)$ into another domain with authorized public key $pk_\sigma \in \mathbf{CER}_\sigma$ according to the following steps.

Step-0(@CP&CSP): Use vk_{KGC} as the input of verifiy to determine whether both $\mathbf{CER}_a$ and $\mathbf{CER}_\sigma$ are valid, and $[x]_{pk_a} \in AD$ in $\mathbf{CER}_a$. If invalid, then output $\perp$; otherwise, execute the following.

Step-1(@CP): Select a random number $r \in \mathbb{Z}_N$, calculate $X_1 \leftarrow [x + r]_{pk_a} = [x]_{pk_a} \cdot [r]_{pk_a}$, and $X_1' \leftarrow \mathtt{PD1}_{SK^{(1)}}(X_1)$, and send X_1 and X_1' to CSP.

Step-2(@CSP): Use PD2 of DT-PKC (see the supplemental material, available online section) to decrypt X_1 and X_1', and obtain $h = x + r$. Then, encrypt h and send $D_1 \leftarrow [h]_{pk_\sigma}$ to the CP.

Step-3(@CP): Calculate the final result as $[z]_{pk_\sigma} \leftarrow D_1 \cdot ([r]_{pk_\sigma})^{N-1}$, where it can be checked $z = x$.

Similarly, the SDT can be used for transforming the computed result $[x]_{pk_\sigma}$ from domain σ to user's domain a (with output $[x]_{pk_a}$), if both $\mathbf{CER}_a$ and $\mathbf{CER}_\sigma$ are valid, and $[x]_{pk_\sigma} \in AD$ in $\mathbf{CER}_\sigma$. Note that system adversaries $\mathcal{A}_1^*$, $\mathcal{A}_2^*$, and $\mathcal{A}_3^*$ in Section 3.2 cannot directly use the SDT to transform some arbitrary ciphertexts into their own domains. It is because these adversaries cannot 1) compromise both CP and CSP concurrently, and 2) generate two fake certificates, and let the target ciphertexts belong to AD in $\mathbf{CER}_\sigma$, as strong unforgeable signatures are used for generating these certificates.

4.3 Secure Computation Components

In this section, we will show how to normalize data before encryption and how to achieve secure computation over normalized encryption. Note that all the following ciphertexts are associated with the same public key. Thus, we omit the subscript operation of the ciphertexts and use $[x]$ to denote the encryption of x.

4.3.1 Data Normalization

The DT-PKC can only encrypt integer value, and SVM requires the storing and processing of non-integer values. To solve the problem, we normalize all data into a fraction before encryption. i.e., the number x^* can be stored as two ciphertexts $\langle x^* \rangle = ([x^+], [x^-])$, such that $x^* \approx \frac{x^+}{x^-}$, where $x^- > 0$.

1. As the DT-PKC is an additive homomorphic encryption scheme, multiplication homomorphic operations on encrypted data cannot be manipulated in a single server. If both CP and CSP collude, the CP and CSP can be considered as one party, and all data stored in the CP can be decrypted. In this situation, only the fully homomorphic encryption scheme can be used for secure computation, which is rather inefficient for homomorphic operations in existing schemes.

2. Specially, if σ^* contains only one party i, then $PT_{\sigma^*} = PT_i$. Access domain AD defines which domain/type of encrypted data can authorize user accesses.

4.3.2 Secure Fraction Computation Protocols

Here, we will construct the secure fraction computation protocols as the basis of our POD.

Secure Fraction Absolute Value Protocol (FAbs): Given $\langle x \rangle$, and it outputs $\langle z \rangle$, such that $z = |x|$. Note that the sign of the fraction is stored in the numerator. The protocol works as follows: $[h] \leftarrow \texttt{SLT}([x^+], [0])$; $[h_1] \leftarrow [1] \cdot [h]^{N-2} = [1-2h]$; $[z^+] \leftarrow \texttt{SM}([x^+], [h_1])$; $[z^-] \leftarrow [x^-]$. Here, we denote $\langle z \rangle \leftarrow \texttt{FAbs}(\langle x \rangle)$.

Secure Ciphertext Oblivious Selection (COS): Given $[x]$, $[y]$ and $[h]$, the COS securely selects x or y and outputs $[z]$, according to h (i.e., $z \leftarrow x$ if $h = 0$ and $z \leftarrow y$ if $h = 1$). The COS is constructed as follows: $[p_1] \leftarrow \texttt{SM}([x^+], [1] \cdot [h]^{N-1})$; $[p_2] \leftarrow \texttt{SM}([y^+], [h])$; $[z] \leftarrow [p_1] \cdot [p_2] = [x \cdot (1-h) + y \cdot h]$. We denote the algorithm as $[z] \leftarrow \texttt{COS}([x], [y], [h])$.

Secure Fraction Oblivious Selection (FOS): Given $\langle x \rangle$ and $\langle y \rangle$ and $[h]$, FOS securely selects x or y and outputs $\langle z \rangle$, according to h (i.e., $z \leftarrow x$ if $h = 0$ and $z \leftarrow y$ if $h = 1$). The COS is constructed as follows:

$$[z^+] \leftarrow \texttt{COS}([x^+], [y^+], [h]); [z^-] \leftarrow \texttt{COS}([x^-], [y^-], [h]),$$

and we denote the algorithm as $\langle z \rangle \leftarrow \texttt{FOS}(\langle x \rangle, \langle y \rangle, [h])$.

Secure Tuple Maximum Protocol (MAX): Given $U_1 = (\langle t_1 \rangle, \langle x_1 \rangle, \langle y_1 \rangle, [z_1], [ID_1])$ and $U_2 = (\langle t_2 \rangle, \langle x_2 \rangle, \langle y_2 \rangle, [z_2], [ID_2])$, such that it outputs $U = (\langle t \rangle, \langle x \rangle, \langle y \rangle, [z], [ID])$, where $t = max(t_1, t_2)$, and x, y, z and ID are t's corresponding values and identity. (1) Compute $[h] \leftarrow \texttt{FCmp}(\langle t_1 \rangle, \langle t_2 \rangle)$. (2) Compute $\langle t \rangle \leftarrow \texttt{FOS}(\langle t_1 \rangle, \langle t_2 \rangle, [h])$; $\langle x \rangle \leftarrow \texttt{FOS}(\langle x_1 \rangle, \langle x_2 \rangle, [h])$; $\langle y \rangle \leftarrow \texttt{FOS}(\langle y_1 \rangle, \langle y_2 \rangle, [h])$; $[z] \leftarrow \texttt{COS}([z_1], [z_2], [h])$; $[ID] \leftarrow \texttt{COS}([ID_1], [ID_2], [h])$. The algorithm is denoted as $U \leftarrow \texttt{MAX}(U_1, U_2)$.

Secure Tuple TOP-1 Protocol (TOP1): Input $U_1, \ldots, U_\delta$, where $U_i = (\langle t_i \rangle, \langle x_i \rangle, \langle y_i \rangle, [z_i], [ID_i])$, $i = 1, \ldots, \delta$. TOP1 outputs $(\langle x' \rangle, \langle y' \rangle, [z'], [ID'])$, where $t' = max(t_1, t_2, \ldots, t_\delta)$, and x', y', z' and ID' are the corresponding values and identity of t'. (1) Place $U_1, \ldots, U_\delta$ into a set Γ, and let $\mathcal{S}(\Gamma)$ be the size of set Γ. (2) The following procedure executes recurrently until only one tuple remains in Γ, i.e., $\mathcal{S}(\Gamma) = 1$. Let the part of U_1 (i.e., $(\langle x_1 \rangle, \langle y_1 \rangle, [z_1], [ID_1])$ and $U^* \leftarrow (\langle x' \rangle, \langle y' \rangle, [z'], [ID'])$) be the final output. If $\mathcal{S}(\Gamma) > 1$, then TOP1 executes as follows: 1) If the size $\mathcal{S}(\Gamma) \bmod 2 = 0$, then calculate $U'_i \leftarrow \texttt{MAX}(U_{2i-1}, U_{2i})$ for $i = 1, \ldots, \mathcal{S}(\Gamma)/2$. Put $U'_1, \ldots, U'_{\mathcal{S}(\Gamma)/2}$ in to a set Γ' and replace Γ by Γ'; 2) If the size $\mathcal{S}(\Gamma) \bmod 2 \neq 0$, then calculate $U'_i \leftarrow \texttt{MAX}(U_{2i-1}, U_{2i})$ for $i = 1, \ldots, (\mathcal{S}(\Gamma) - 1)/2$. Put $U'_1, \ldots, U'_{(\mathcal{S}(\Gamma)-1)/2}, U_{\mathcal{S}(\Gamma)}$ in a set Γ' and replace Γ by Γ'. We denote the TOP1 as $U^* \leftarrow \texttt{TOP1}(U_1, \ldots, U_\delta)$.

4.4 Secure Fraction Approximation Protocol (FApx)

As all data are encrypted, the plaintext length of the ciphertext may easily overflow when a large number of secure computation are involved in the SVM training and classification phase. Although a secure data approximate method was proposed in our previous work [13] to securely reduce the plaintext length, the secure approximate method in [13] may fail to reduce the plaintext length if both numerator and denominator are co-prime. Moreover, the overhead of our previously published approximate method is relatively high, due to the use of the secure division protocol.

Thus, we design a new FApx, which can be used to reduce the plaintext size to overcome the above limitation. FApx is now described below: Given $\langle x \rangle = ([x^+], [x^-])$, where $\mathcal{L}(x^+) = \mu_1$ and $\mathcal{L}(x^-) = \mu_2$. FApx outputs $\langle z \rangle = ([z^+], [z^-])$, where $\mathcal{L}(z^+) = \mu_1 - \kappa$ and $\mathcal{L}(z^-) = \mu_2 - \kappa$, and the construction is as follows.

(1) Extract $[x^+]$'s absolute value $[x']$ and its corresponding symbol $[s_x]$ securely, i.e., $([x'], [s_x]) \leftarrow \texttt{SSBA}([x^+])$;
(2) Use SBD to securely obtain the bit representations of $[\mathbf{x}']$ and $[\bar{x}^-]$, i.e., $([\mathbf{x}^+_{\mu_1-1}], \ldots, [\mathbf{x}^+_0]) \leftarrow \texttt{SBD}([x'])$ and $([\mathbf{x}^-_{\mu_1-1}], \ldots, [\mathbf{x}^-_0]) \leftarrow \texttt{SBD}([x^-])$;
(3) Finally, calculate $[z'] \leftarrow [\mathbf{x}^+_\kappa] \cdot [\mathbf{x}^+_{\kappa+1}]^2 \cdots [\mathbf{x}^+_{\mu_1-1}]^{2^{\mu_1-\kappa}}$, and obtain the approximation result as $[z^+] \leftarrow \texttt{SM}([z'], [s_x]^2 \cdot [1]^{N-1})$; $[z^-] \leftarrow [\mathbf{x}^-_\kappa] \cdot [\mathbf{x}^-_{\kappa+1}]^2 \cdots [\mathbf{x}^-_{\mu_1-1}]^{2^{\mu_1-\kappa}}$.

When the calculated plaintext reaches its upper bound, the system can automatically call FApx. Due to page limitations, we will not specifically explain the usage of FApx in the following section. Here, we give an example to demonstrate the correctness of FApx.

Example of FApx: Given $\langle x \rangle = ([-92], [99])$, where $\|x^+\| = \|x^-\| = 7$, and we wish to reduce the plaintext length of x^+ and x^- to 5 and obtain a new fraction result $\langle z \rangle$. First, we use SSBA to obtain $[92]$ and symbol $[0]$. Next, we use SBD to expand $[92]$ into $([1], [0], [1], [1], [1], [0], [0])$, and expand $[99]$ into $([1], [1], [0], [0], [0], [1], [1])$. Then, we calculate $[z'] \leftarrow [1] \cdot [1]^2 \cdot [1]^4 \cdot [0]^8 \cdot [1]^{16} = [23]$ and $[z^-] \leftarrow [0] \cdot [0]^2 \cdot [0]^4 \cdot [1]^8 \cdot [1]^{16} = [24]$. Finally, we calculate $[s'] \leftarrow [0]^2 \cdot [-1] = [-1]$ and $[-23] = [z^+] \leftarrow \texttt{SM}([23]; [s'])$.

By executing FApx, we can approximate the original fraction $\frac{-92}{99} \approx -0.929$ into a new faction $\frac{-23}{24} \approx -0.958$.

5 Proposed POD Framework

Before using privacy-preserving SVM for decision-making, we need to train the encrypted SVM before usage. Note that the SVM can be formulated as an optimization problem (see Section 2.1). The goal is to find appropriate $\alpha^*_1, \ldots, \alpha^*_n$ to satisfy the SVM dual problem, which requires the solution of a very large quadratic programming (QP) optimization problem. The Sequential Minimal Optimization (SMO) algorithm is an efficient way of solving the dual problem due to the derivation of the SVM, which breaks the large QP problem into a series of smallest possible QP problems (see [16] for more information). These small QP problems are solved analytically. In this section, we will discuss how to securely select α_1 and α_2 from $\alpha^*_1, \ldots, \alpha^*_n$, and use SMO algorithm to securely refresh α_1 and α_2.

To successfully achieve computer-aided drug design, one accepted basic tenet is that similar molecules have similar activities. To allow the server to automatically achieve the drug design, we transform chemical information into a useful number or the result of some standardized experiment which can be used for logic and mathematical procedure. The transformed information is known as the molecular descriptors. There are a numbers of ways to describe molecular [17], and in this paper, we use a binary vector $\vec{x}_i = (x_{i,1}, \ldots, x_{i,t})$ encoding chemical substructures (or fragments), and each bit records the presence ("1") or absence ("0") of a fragment in the molecule. Moreover, we use a bit y to indicate active ("1") and non-active ("0") to represent the chemical structure (see Fig. 3).

5.1 Overview of POD

As we have previously explained, (for ease of understanding) in this paper each DP $\mathscr{D}_i$ contains only one drug formula. Here, we give the basic overview of POD.

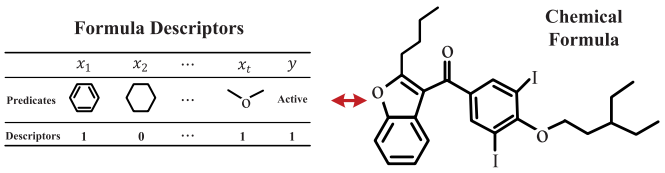


Fig. 3. Chemical formula description.

Secure Drug Formula Outsourcing Phase. To minimize the storage space, each DP $\mathscr{D}_i$'s drug formula descriptors $\vec{x}_i = (x_{i,1}, \ldots, x_{i,t})$ and decision y_i are encrypted with the DP's own public key, and formed as $[\vec{x}_i]_{pk_i} = ([x_{i,1}]_{pk_i}, \ldots, [x_{i,t}]_{pk_i})$ and $[y_i]_{pk_i}$, prior to outsourcing to the CP for storage. Also, the AP initializes the SVM parameters $\alpha_1^*, \ldots, \alpha_n^*, \beta^*$, uses his/her own public key $pk_{\mathscr{A}}$ to encrypt and obtain $\langle \alpha_1^* \rangle_{pk_{\mathscr{A}}}, \ldots, \langle \alpha_n^* \rangle_{pk_{\mathscr{A}}}, \langle \beta^* \rangle_{pk_{\mathscr{A}}}$, and sends them to the CP for storage.

Secure Drug Formula Pre-Process Phase. Before training the SVM, the CP needs to decide which DP's data can be manipulated by AP $\mathscr{A}$, i.e., uses Step-0 of SDT to determine whether $\mathbf{CER}_{\mathscr{A}}$ and $\mathbf{CER}_{\sigma}$ are valid, where $\sigma = \{\mathscr{D}_1, \ldots, \mathscr{D}_n : \mathscr{A}\}$. If this is validated, then ciphertexts $[\vec{x}_i]_{pk_i}$ from $\mathscr{D}_i$ and SVM parameters $\langle \alpha_1^* \rangle_{pk_{\mathscr{A}}}, \ldots, \langle \alpha_n^* \rangle_{pk_{\mathscr{A}}}, \langle \beta^* \rangle_{pk_{\mathscr{A}}}$ will be transformed into domain σ using Steps 1 to 3 of SDT. Since all data belong to the same domain pk_σ, for simplicity, we omit the notation pk_σ in $[x_i]_{pk_\sigma}$ and $\langle \alpha_i^* \rangle_{pk_\sigma}$, and use $[x]$ and $\langle \alpha^* \rangle$ instead. Moreover, both CP and CSP calculate all kernel functions between every two instances in the training set offline, i.e., calculate $[K_{ij}^*] \leftarrow \texttt{SKer}([\vec{x}_i], [\vec{x}_j])$ for $i, j = 1, \ldots, n$. Moreover, we use STyp2 to compute $\langle E_i^* \rangle \leftarrow \texttt{STyp2}([\vec{x}_i], S^*)$, where $S^* = \{([\vec{x}_1], [y_1]), \ldots, ([\vec{x}_n], [y_n]), \langle \alpha_1^* \rangle, \ldots, \langle \alpha_n^* \rangle, \langle \beta^* \rangle\}$ (see Section 5.2 for construction of Sker and STyp2).

Secure SVM Training with Encrypted Drug Descriptors. To refresh all the parameters in SVM, CP selects $\langle \alpha_1 \rangle$ and $\langle \alpha_2 \rangle$ from $\langle \alpha_1^* \rangle, \ldots, \langle \alpha_n^* \rangle$, such that α_1 and α_2 are the first two parameters in the above parameters that do not satisfy the Karush-Kuhn-Tucker (KKT) condition (see Section 5.3). To train the encrypted SVM, Secure Sequential Minimal Optimization (SSMO) is designed to refresh α_1 and α_2 in a privacy-preserving way (see Section 5.4). As the newly generated $\alpha_1^{(e)}$ and $\alpha_2^{(e)}$ are encrypted, CP needs to find the corresponding positions in S^*, and refreshes them with the new encrypted values (see Section 5.5). Finally, we securely check to determine whether the training phase of SVM has completed (see Section 5.6). Since all parameters are encrypted, it is not possible for the CP to decide when to stop the training.

Secure Outsourced SVM Classification for Unknown Drug Molecules. Once secure drug classification is needed, DT $\mathscr{B}$ encrypts and uploads $[\vec{x}_b]_{pk_{\mathscr{B}}}$ to the CP. Next, the encrypted $[\vec{x}_b]_{pk_{\mathscr{B}}}$ are transformed into domain $\sigma' = \{\mathscr{D}_1, \ldots, \mathscr{D}_n, \mathscr{A} : \mathscr{B}\}$, and then CP performs the secure SVM classification on-the-fly. Finally, the encrypted result is transformed into domain $\mathscr{B}$ and sends back $\mathscr{B}$ for decryption (see Section 5.7).

5.2 Secure Function Computation Protocols

A kernel function is a similarity metric between the input objects, which is considered as key component of SVM. In practice, a few kernels (such as Polynomial Kernel, Gaussian Kernel, Sigmoid Kernel) would be more appropriate for most common settings. In the proposed POD, we choose polynomial kernel $K(\vec{x}, \vec{x}') = (1 + \vec{x} \cdot \vec{x}')^d$ as the basis of SVM. For privacy classification and training, we also need to securely calculate two kinds of functions: Type-1 function $g_1(\vec{x}) = \sum_{j=1}^{n} \alpha_j y_j K(\vec{x}, \vec{x}_j) + \beta$ and Type-2 function $g_2(\vec{x}, y) = \sum_{j=1}^{n} \alpha_j y_j K(\vec{x}, \vec{x}_j) + \beta - y$.

Secure Kernel Function Computation Protocol (SKer): Input $[\vec{x}] = ([x_1], \ldots, [x_t])$ and $[\vec{x}'] = ([x'_1], \ldots, [x'_t])$, SKer securely outputs $[K(\vec{x}, \vec{x}')]$ as follows: (1) calculate encrypted $h_2 = 1 + \vec{x} \cdot \vec{x}'$ by $[h_1] \leftarrow \texttt{SIP}([x_1], \ldots, [x_t]; [x'_1], \ldots, [x'_t])$ and $[h_2] \leftarrow [1] \cdot [h_1] = [1 + h_1]$. (2) compute encrypted $(h_2)^d$ with exponent d (binary representation as $(\mathfrak{d}_{\mu-1}, \ldots, \mathfrak{d}_0)$), we use the square-and-multiply idea taht executes as follows: initialize $[z] \leftarrow [x]$. Then for $i = \mu - 1$ to 0, if $\mathfrak{d}_i = 1$, calculate $[z] \leftarrow \texttt{SM}([z], [x])$; otherwise, complete and jump to the beginning of the loop for the next iteration.

Secure Type-1 Function Computation (STyp1): Input $[\vec{x}]$ and $S^* = \{([\vec{x}_1], [y_1]), \ldots, ([\vec{x}_n], [y_n]), \langle \alpha_1 \rangle, \ldots, \langle \alpha_n \rangle, \langle \beta \rangle\}$, STyp1 securely calculates Type-1 function and outputs $\langle g_1(\vec{x}) \rangle$ as follows: (1) Initialize $[g_1(\vec{x})^+] \leftarrow [0]$ and $[g_1(\vec{x})^-] \leftarrow [0]$, respectively. (2) The goal of this step is to calculate the encrypted $L_j = y_j K(\vec{x}, \vec{x}_j)$, i.e., for $j = 1, \ldots, n$, calculate $[T'_j] \leftarrow \texttt{SKer}([\vec{x}], [\vec{x}_j])$ and $[l_j] \leftarrow \texttt{SM}([T'_j], [y_j])$. Note that $[l_j]$ can be considered an encrypted fraction $\langle L \rangle$, where $[L_j^+] = [l_j]$ and $[L_j^-] = [1]$. (3) The final $g_1(\vec{x})$ can be calculated by $\langle o \rangle \leftarrow \texttt{FIP}(\langle \alpha_1 \rangle, \ldots, \langle \alpha_n \rangle; \langle L_1 \rangle, \ldots, \langle L_n \rangle)$ and $\langle g_1(\vec{x}) \rangle \leftarrow \texttt{FAdd}(\langle o \rangle, \langle \beta \rangle)$. Here, we denote STyp1 as $\langle g_1(\vec{x}) \rangle \leftarrow \texttt{STyp1}([\vec{x}], S^*)$.

Secure Type-2 Function Computation (STyp2): Similar to the input of STyp1, STyp2 securely calculates kernel function and outputs $\langle g_2(\vec{x}) \rangle$ as follows: (1) Calculate $\langle G \rangle \leftarrow \texttt{STyp1}([\vec{x}], S^*)$. (2) Compute $[X] \leftarrow \texttt{SM}([G^-], [y])$; $[g_2(\vec{x})^+] \leftarrow [G^+] \cdot [X]^{N-1}$; $[g_2(\vec{x})^-] \leftarrow [G^-]$. Here, we denote the protocol as $\langle g_2(\vec{x}) \rangle \leftarrow \texttt{STyp2}([\vec{x}], S^*)$.

5.3 Secure Parameter Selection

To refresh all α_i^* ($i = 1$ to n) in the SVM, CP is required to securely select α_1 and α_2 for the SVM parameters, i.e., find first α_i^* (denote α_1, the corresponding E_i denotes as E_1) that does not satisfy the KKT condition.[3] Once α_1 is selected, CP tests all α_i^* to calculate $T_i^* = |E_1 - E_i^*|$, finds α_2 such that $T_2 = max(T_1^*, \ldots, T_n^*)$, where $\langle E_i^* \rangle \leftarrow \texttt{STyp2}([\vec{x}_i], S^*)$ and $S^* = \{([\vec{x}_1], [y_1]), \ldots, ([\vec{x}_n], [y_n])\}$. Next, we will show how to construct a protocol to check whether an element α_i^* does not satisfy the KKT condition securely (SKKT) and securely select α_1 and α_2 among all the encrypted α_i^*.

5.3.1 Secure KKT Check Protocol (SKKT)

Given $\langle \alpha \rangle, \langle E \rangle$ and $[y]$, SKKT outputs the encrypted element $[f]$ to determine whether α satisfies the KKT condition, i.e., if α does not satisfy KKT, then $f = 1$; otherwise, $f = 0$.

(1) As the comparison involved in the KKT condition is strictly less than comparison, CP constructs four encrypted $[F_1], [F_2], [F'_1]$ and $[F'_2]$ to check whether $\alpha < C$, $yE < -\varepsilon$, $C < \alpha$ and $\varepsilon < yE$ hold, respectively, i.e., compute $[Z^+] \leftarrow \texttt{SM}([E^+], [y])$; $[Z^-] \leftarrow [E^-]$; $[F_1] \leftarrow \texttt{FCmp}(\langle \alpha \rangle, \langle C \rangle)$; $[F_2] \leftarrow \texttt{FCmp}(\langle Z \rangle, \langle -\varepsilon \rangle)$; $[F'_1] \leftarrow \texttt{FCmp}(\langle C \rangle, \langle \alpha \rangle)$; $[F'_2] \leftarrow \texttt{FCmp}(\langle \varepsilon \rangle, \langle Z \rangle)$.

3. For $\alpha_1^*, \cdots, \alpha_n^*$, α_τ is the first not to satisfy the KKT condition ($\alpha_\tau < C$ and $y_\tau E_\tau < -\varepsilon$; or $\alpha_\tau > C$ and $y_\tau E_\tau > \varepsilon$), where $E_\tau = g_2(\vec{x}_\tau)$, C is the upper bound of α_τ, ε is the margin and typically set to 10^{-3}.

(2) With the above four ciphertexts, we can construct the final output by $f = (F_1 \wedge F_2) \vee (F_1' \wedge F_2')$. This logic can be constructed by $[U_1] \leftarrow \texttt{SM}([F_1], [F_2])$; $[U_2] \leftarrow \texttt{SM}([F_1'], [F_2'])$; $[f_1] \leftarrow \texttt{SM}([1] \cdot [U_1]^{N-1}, [U_2])$; $[f_2] \leftarrow \texttt{SM}([U_1], [1] \cdot [U_2]^{N-1})$, and outputs $[f] \leftarrow \texttt{SM}([f_1], [f_2])$.

5.3.2 Secure α_1 and α_2 Selection Protocol (SSel)

To successfully find α_1 and α_2, the idea of SSel is to first construct the encrypted tuple $U_i^* = (\langle \alpha_i^* \rangle, \langle E_i^* \rangle, [y_i^*], [ID_i^*])$, where $ID_i^* \neq 0$ is the unique identity associated with α_i^*. Next, we use SKKT to construct the encrypted s_i' to indicate whether α_i^* is the first element that does not satisfy the KKT condition, where v_{i-1} is used to record that no element does not satisfy KKT condition previously if $v_{i-1} = 1$. The tuple $(\langle \alpha_1 \rangle, \langle E_1 \rangle, [y_1], [ID_1])$ can be easily constructed as only one element is equal to 1 among $s_1', \ldots, s_n'$. Once the encrypted α_1 is generated, we can construct element $T_i^* \leftarrow (\langle t_i^* \rangle, \langle \alpha_i^* \rangle, \langle E_i^* \rangle, [y_i^*], [ID_i^*])$, where $t_i^* = |E_1 - E_i^*|$, and use TOP-1 to find the tuple $(\langle \alpha_2'' \rangle, \langle E_2'' \rangle, [y_2''], [ID_2''])$ with maximum t_i^*, which is the tuple that contains encrypted element α_2. The detailed construction of SSel can be found in Algorithm 1 and described as follows.

Algorithm 1. Secure α_1 and α_2 Selection Protocol

Input: $U_i^* = (\langle \alpha_i^* \rangle, \langle E_i^* \rangle, [y_i^*], [ID_i^*])$ for $i = 1, \ldots, n$.
Output: $(\langle \alpha_1 \rangle, \langle E_1 \rangle, [y_1], [ID_1])$; $(\langle \alpha_2 \rangle, \langle E_2 \rangle, [y_2], [ID_2])$.

1: Initialize $[v_0] \leftarrow [1]$.;
2: **for** $i = 1, \ldots, n$ **do**
3: $[s_i] \leftarrow \texttt{SKKT}(\langle \alpha_i^* \rangle, \langle E_i^* \rangle, [y_i^*])$;
4: $[v_i] \leftarrow \texttt{SM}([v_{i-1}], [1] \cdot [s_i]^{N-1})$; $[s_i'] \leftarrow \texttt{SM}([v_{i-1}], [s_i])$;
5: $[(\alpha_i')^+] \leftarrow \texttt{SM}(([\alpha_i^*)^+], [s_i'])$; $[(\alpha_i')^-] \leftarrow \texttt{SM}(([\alpha_i^*)^-], [s_i'])$;
6: $[(E_i')^+] \leftarrow \texttt{SM}([(E_i^*)^+], [s_i'])$; $[(E_i')^-] \leftarrow \texttt{SM}([(E_i^*)^-], [s_i'])$;
7: $[y_i'] \leftarrow \texttt{SM}([y_i^*], [s_i'])$; $[ID_i'] \leftarrow \texttt{SM}([ID_i^*], [s_i'])$;
8: Multiply all $[(\alpha_i)^+]$ to generate $[(\alpha_1)^+]$. Similarly, generate $[\alpha_1^-]$, $[E_1^+]$, $[E_1^-][y_1]$, $[ID_1]$ by multiplying $[(\alpha_i')^-]$, $[(E_i')^+]$, $[(E_i')^-]$, $[y_i']$, $[ID_i']$, respectively.
9: **for** $i = 1, \ldots, n$ **do**
10: $\langle t_i^* \rangle \leftarrow \texttt{FAbs}(\texttt{FSub}(\langle E_1 \rangle, \langle E_i^* \rangle))$.
11: Denote $T_i^* \leftarrow (\langle t_i^* \rangle, \langle \alpha_i^* \rangle, \langle E_i^* \rangle, [y_i^*], [ID_i^*])$;
12: $(\langle \alpha_2'' \rangle, \langle E_2'' \rangle, [y_2''], [ID_2'']) \leftarrow \texttt{TOP1}(T_1^*, \ldots, T_n^*)$;
13: $\langle \alpha_2 \rangle \leftarrow \texttt{FOS}(\langle \alpha_2'' \rangle, \langle 0 \rangle, [v_n])$; $\langle E_2 \rangle \leftarrow \texttt{FOS}(\langle E_2'' \rangle, \langle 0 \rangle, [v_n])$;
14: $[y_2] \leftarrow \texttt{COS}([y_2''], [0], [v_n])$; $[ID_2] \leftarrow \texttt{COS}([ID_2''], [0], [v_n])$.
15: **Return** $(\langle \alpha_1 \rangle, \langle E_1 \rangle, [y_1], [ID_1])$; $(\langle \alpha_2 \rangle, \langle E_2 \rangle, [y_2], [ID_2])$.

In line 1, initialize $[v_0] \leftarrow [1]$. Then, for each input tuple $U_1, \ldots, U_n$, use SKKT to test whether the $\langle \alpha_i^* \rangle$ in U_i^* satisfies the KKT conditions or not (line 3). If α_i^* is the first element that does not satisfy the KKT condition, then calculate $v_i = 0$ and $s_i' = 1$ using SM protocol (line 4), and obtain the corresponding value $\langle \alpha_i' \rangle$, $\langle E_i' \rangle$, $[y_i']$ and $[ID_i']$, respectively (line 5-7). As only the first tuple that does not satisfy KKT contains encrypted non-zero value and other tuples are encrypted zeros, use DT-PKC homomorphic addition to obtain $\langle \alpha_1 \rangle$, $\langle E_1 \rangle$, $[y_1]$ and $[ID_1]$ (line 8). Next, use FAbs and FSub to compute $\langle T_i^* \rangle \leftarrow \langle |E_i - E_i^*| \rangle$ (lines 9-10), and use the TOP1 to obtain $\langle \alpha_2'' \rangle$, $\langle E_2'' \rangle$, $[y_2'']$ and $[ID_2'']$ (line 11). If there exists a α_1 that does not satisfy the KKT condition, then output $U_2 \leftarrow (\langle \alpha_2'' \rangle, \langle E_2'' \rangle, [y_2''], [ID_2''])$. Otherwise, output $U_2 = U_1 \leftarrow (\langle 0 \rangle, \langle 0 \rangle, [0], [0])$, where the numerator of $\langle 0 \rangle$ is equal to [0], and the denominator of $\langle 0 \rangle$ is equal to [1]. Note that $ID_1 = ID_2 = 0 \notin \{ID_1^*, \ldots, ID_n^*\}$ if no α_i^* $(i = 1, \ldots, n)$ does not satisfy KKT condition (lines 13-14).

5.4 Secure Sequential Minimal Optimization (SSMO)

After executing Ssel, we can securely obtain $U_1 = (\langle \alpha_1 \rangle, \langle E_1 \rangle, [y_1], [ID_1])$ and $U_2 = (\langle \alpha_2 \rangle, \langle E_2 \rangle, [y_2], [ID_2])$. We denote them as $(\langle \alpha_1^{(o)} \rangle, \langle E_1^{(o)} \rangle, [y_1^{(o)}], [ID_1^{(o)}])$ and $(\langle \alpha_2^{(o)} \rangle, \langle E_2^{(o)} \rangle, [y_2^{(o)}], [ID_2^{(o)}])$, respectively. As the SMO algorithm is an efficient way of solving the resultant dual problem, we will construct its corresponding encrypted version—Secure Sequential Minimal Optimization (SSMO)—to securely refresh $\langle \alpha_1^{(o)} \rangle$, $\langle \alpha_2^{(o)} \rangle$, and $\langle \beta^{(o)} \rangle$, and output $\langle \alpha_1^{(e)} \rangle$, $\langle \alpha_2^{(e)} \rangle$, and $\langle \beta^{(e)} \rangle$.

Step-1: Before refreshing $\alpha_1^{(o)}$ and $\alpha_2^{(o)}$, CP computes the lower bound L and upper bound H of these two values, i.e., if $y_1^{(o)} \neq y_2^{(o)}$, then calculates $L \leftarrow max(0, \alpha_2^{(o)} - \alpha_1^{(o)})$, $H \leftarrow min(C, C + \alpha_2^{(o)} - \alpha_1^{(o)})$; if $y_1^{(o)} = y_2^{(o)}$, then calculates $L \leftarrow max(0, \alpha_2^{(o)} + \alpha_1^{(o)} - C)$, $H \leftarrow min(C, \alpha_2^{(o)} + \alpha_1^{(o)})$. It can be securely computed as follows: First, calculate $\langle h_1 \rangle \leftarrow \langle max(0, \alpha_2^{(o)} - \alpha_1^{(o)}) \rangle$ and $\langle h_2 \rangle \leftarrow \langle min(C, C + \alpha_2^{(o)} - \alpha_1^{(o)}) \rangle$ and $\langle h_3 \rangle \leftarrow \langle max(0, \alpha_2^{(o)} + \alpha_1^{(o)} - C) \rangle$ and $\langle h_4 \rangle \leftarrow \langle min(C, \alpha_2^{(o)} + \alpha_1^{(o)}) \rangle$ with the combination of FSub, FAdd, FCmp and FOS. Second, generate $\langle h_e \rangle \leftarrow \texttt{SEQ}([y_1^{(o)}], [y_2^{(o)}])$ to determine whether $y_1^{(o)}$ and $y_2^{(o)}$ are equal. Then, FOS is used to compute $\langle L \rangle$ using $\langle h_3 \rangle, \langle h_1 \rangle, [h_e]$ and calculate $\langle H \rangle$ with $\langle h_4 \rangle, \langle h_2 \rangle, [h_e]$.

Step-2: In this step, calculate the encrypted new unbounded value $\langle \alpha_2^{(e,u)} \rangle$ associated with $\alpha_2^{(o)}$, such that $\alpha_2^{(e,u)} = \alpha_2^{(o)} + \frac{y_2(E_1^{(o)} - E_2^{(o)})}{\eta}$, where $\eta = K_{11}^{(o)} + K_{22}^{(o)} - 2K_{12}^{(o)}$. Since all data are encrypted, it impossible to know the value of $K_{11}^{(o)}$, $K_{12}^{(o)}$, $K_{22}^{(o)}$. Thus, $ID_1^{(o)}$ and $ID_2^{(o)}$ are used to locate the position, such that 1) if $ID_1^{(o)} = ID_i^*$, let $K_{11}^{(o)} \leftarrow K_{ii}^*$, 2) if $ID_2^{(o)} = ID_i^*$, let $K_{22}^{(o)} \leftarrow K_{ii}^*$, 3) if $ID_1^{(o)} = ID_i^*$ and $ID^{(o)} = ID_j^*$, let $K_{12}^{(o)} \leftarrow K_{ij}^*$; otherwise, let $K_{11}^{(o)}, K_{22}^{(o)}, K_{12}^{(o)}$ be [0], where K_{ij}^* can be computed offline (see Section 5.1). Note that all the identity are encrypted in the integer format, we can directly use the secure integer protocol for secure computation. Finally, calculate $\langle \eta \rangle$ and $\langle \alpha_2^{(e,u)} \rangle$ using FSub, FMul and FAdd.

Step-3: Reduce $\alpha_2^{(e,u)}$ according to the upper and lower bounds, and hence obtain $\langle \alpha_2^{(e)} \rangle$, such that

$$\alpha_2^{(e)} = \begin{cases} H, & \alpha_2^{(e,u)} > H \\ \alpha_2^{(e,u)}, & L \leq \alpha_2^{(e,u)} \leq H \\ L, & \alpha_2^{(e,u)} < L. \end{cases} \quad (1)$$

Note that $\langle \alpha_2^{(e)} \rangle$ can be easily computed according to $\langle \alpha_2^{(e,u)} \rangle$, $\langle H \rangle$ and $\langle L \rangle$ with secure fraction protocols. Once we obtain $\langle \alpha_2^{(e)} \rangle$, we can calculate the encrypted

$\alpha_1^{(e)} = \alpha_1^{(o)} + y_1^{(o)} \cdot y_2^{(o)} \cdot (\alpha_2^{(o)} - \alpha_2^{(e)})$ to refresh the corresponding encrypted $\alpha_1^{(o)}$.

Step-4: To generate new encrypted value $\beta^{(e)}$, calculate encrypted $\langle \beta_1^{(e)} \rangle$ and $\langle \beta_2^{(e)} \rangle$ with FAdd, FSub and FMul, such that

$$\beta_1^{(e)} = -E_1^{(o)} - y_1^{(o)} K_{11}^{(o)} (\alpha_1^{(e)} - \alpha_1^{(o)}) - y_2 K_{21}^{(o)} (\alpha_2^{(e)} - \alpha_2^{(o)}) + \beta^{(o)},$$

$$\beta_2^{(e)} = -E_2^{(o)} - y_1^{(o)} K_{12}^{(o)} (\alpha_1^{(e)} - \alpha_1^{(o)}) - y_2 K_{22}^{(o)} (\alpha_2^{(e)} - \alpha_2^{(o)}) + \beta^{(o)}.$$

Then, generate the new encrypted value $\langle \beta^{(e)} \rangle$ according to $\langle \beta_1^{(e)} \rangle$ and $\langle \beta_2^{(e)} \rangle$, i.e., if $0 < \alpha_1^{(e)} < C$, then $\beta^{(e)} = \beta_1^{(e)}$; if $0 < \alpha_2^{(e)} < C$, then $\beta^{(e)} = \beta_2^{(e)}$; otherwise, $\beta^{(e)} = \frac{\beta_1^{(e)} + \beta_2^{(e)}}{2}$. It can be achieved by the combination of FAdd, FCmp, FDiv and FOS protocols.

5.5 Secure Parameter Update (SPU)

After running SSMO and obtaining $\alpha_1^{(o)}$ and $\alpha_2^{(o)}$, CP and CSP construct a protocol called SPU to find the original corresponding values in S^*, and refresh them with the newly generated values $\alpha_1^{(e)}$ and $\alpha_2^{(e)}$, and refresh all E_i using the refreshed $\alpha_1^*, \ldots, \alpha_n^*$.

Step-0 (Process Offline) Calculate $\langle L_{ij} \rangle \leftarrow \langle y_j K_{ij}^* \rangle$ for $i = 1, \ldots, n$ and $j = 1, \ldots, n$ as follows, compute $[L_{ij}^+] \leftarrow \mathtt{SM}([y_i], [(K_{ij}^*)^+])$ and let $[L_{ij}^-] \leftarrow [(K_{ij}^*)^-]$.

Step-1. Use $[ID_1^{(o)}]$ and $[ID_2^{(o)}]$ to securely locate the original position i of $\alpha_1^{(e)}$ and $\alpha_2^{(e)}$ in S^*, and then refresh the value in these two positions with new values $\alpha_1^{(e)}$ and $\alpha_2^{(e)}$. Technically, this step will execute (1)-(3) for $i = 1, \ldots, n$: (1) test whether ID_i^* is equal to $ID_1^{(o)}$ or $ID_2^{(o)}$, $[v_1] \leftarrow \mathtt{SEQ}([ID_i^*], [ID_1^{(o)}])$; $[v_2] \leftarrow \mathtt{SEQ}([ID_i^*], [ID_2^{(o)}])$; (2) use FOS protocol to securely refresh $\langle \alpha_i^* \rangle$, i.e, $\langle \alpha_i'' \rangle \leftarrow \mathtt{FOS}(\langle \alpha_1^{(e)} \rangle, \langle \alpha_i^* \rangle, [v_1])$; $\langle \alpha_i'' \rangle \leftarrow \mathtt{FOS}(\langle \alpha_2^{(e)} \rangle, \langle \alpha_i^* \rangle, [v_2])$; (3) replace $\langle \alpha_i^* \rangle$ by $\langle \alpha_i'' \rangle$.

Step-2. Compute $\langle E_i \rangle$ such that $E_i = \sum_{j=1}^n \alpha_j'' L_{i,j} + \beta^{(e)} - y_i$ for $i = 1, \ldots, n$. This construction is similar to the construction of STyp2. Replace $\langle E_i^* \rangle$ by $\langle E_i \rangle$.

5.6 Remote Training Completion Check

To train the SVM, CP runs secure parameter selection, SSMO and secure parameter update for certain numbers of iterations (e.g., 100 iterations). As all data are encrypted, CP cannot know whether the training phase of SVM has completed or not. Here, we design a new mechanism to allow an authorized remote user AP to determine whether the training phase is completed, i.e., no α_i^* does not satisfy the KKT condition. The mechanism constructs as follows:

Step-1(@CP): Execute $(U_1, U_2) \leftarrow \mathtt{SSel}(U_1^*, \ldots, U_n^*)$, where $U_1 = (\langle \alpha_1 \rangle, \langle E_1 \rangle, [y_1], [ID_1])$ and $U_2 = (\langle \alpha_2 \rangle, \langle E_2 \rangle, [y_2], [ID_2])$. Then, compute $[V] \leftarrow \mathtt{SEQ}([ID_1], [0])$. After that, $[V]$ is transformed into AP's domain with $[V]_{pk_{\mathscr{A}}} \leftarrow \mathtt{SDT}([V], pk_{\mathscr{A}})$ if both $\mathbf{CER}_{\mathscr{A}}$ and $\mathbf{CER}_{\sigma}$ are valid. Otherwise, abort the algorithm and send $\perp$ to $\mathscr{A}$.

Step-2(@AP): Decrypt $[V]_{pk_{\mathscr{A}}}$ with a's own private key $sk_{\mathscr{A}}$. If $V = 0$, then sends CONTINUE to CP; otherwise ($V = 1$), sends STOP to CP.

Step-3(@CP): If CONTINUE is received, then continually execute SSMO to refresh $\langle \alpha_1^* \rangle, \ldots, \langle \alpha_n^* \rangle, \langle \beta^* \rangle$. If STOP is received, then SDT is used to transform $\langle \alpha_1^* \rangle, \ldots, \langle \alpha_n^* \rangle$, $\langle \beta^* \rangle$ into $\langle \alpha_1^* \rangle_{pk_{\mathscr{A}}}, \ldots, \langle \alpha_n^* \rangle_{pk_{\mathscr{A}}}, \langle \beta^* \rangle_{pk_{\mathscr{A}}}$, and sends these transformed results to AP.

5.7 Secure SVM Classification

Once DT $\mathscr{B}$'s encrypted $[\vec{x}_b]_{pk_{\mathscr{B}}}$ are uploaded to the CP, the encrypted trained SVM can be used to perform classification and obtain the result $[y_b]_{\sigma'}$, where $y_b = Sign(\sum_{j=1}^n \alpha_j^* y_i K(\vec{x}_j, \vec{x}_b) + \beta^*)$, σ' is the encrypted domain associated with $\{\mathscr{D}_1, \ldots, \mathscr{D}_n, \mathscr{A} : \mathscr{B}\}$, and $\alpha_1^*, \ldots, \alpha_n^*, \beta^*$ are the SVM parameters. Finally, $[y_b]_{pk_{\sigma'}}$ is transformed into $[y_b]_{pk_{\mathscr{B}}}$ and sent back to $\mathscr{B}$.

Step-0(@CP&CSP): Before computation, transform all the data in S^* into domain σ' using SDT. Note that this step can be performed offline by CP and CSP.

Step-1(@DT $\mathscr{B}$): $\mathscr{B}$ encrypts $\vec{x}_b = (x_{b,1}, \ldots, x_{b,t})$ and obtains $[\vec{x}_b]_{pk_{\mathscr{B}}} = ([x_{b,1}]_{pk_{\mathscr{B}}}, \ldots, [x_{b,t}]_{pk_{\mathscr{B}}})$, and sends $[\vec{x}_b]_{pk_{\mathscr{B}}}$ to CP.

Step-2(@CP): (1) Once encrypted attributes are received, CP transforms the encrypted domain of these ciphertexts by computing $[x_i]_{pk_{\sigma'}} \leftarrow \mathtt{SDT}([x_i]_{pk_{\mathscr{B}}})$ for $i = 1, \ldots, t$, if both $\mathbf{CER}_{\mathscr{B}}$ and $\mathbf{CER}_{\sigma'}$ are valid. Otherwise, abort the algorithm and send $\perp$ to $\mathscr{B}$. Note that all the ciphertexts below are encrypted under $pk_{\sigma'}$. For simplicity, we omit the notation $pk_{\sigma'}$ in $[x]_{pk_{\sigma'}}$ and use $[x]$ instead. (2) Calculate $\langle g_1(\vec{x}_b) \rangle \leftarrow \mathtt{STyp1}([\vec{x}_b], S^*)$, where $S^* = \{([\vec{x}_1], [y_1]), \ldots, ([\vec{x}_n], [y_n]), \langle \alpha_1^* \rangle, \ldots, \langle \alpha_n^* \rangle, \langle \beta^* \rangle\}$. (3) CP needs to look at the symbol of $[g_1(\vec{x}_b)^+]$ and decides on the final result $[y]$, i.e., calculates $[O] \leftarrow \mathtt{SLT}([g_1(\vec{x}_b)^+], [0])$ and $[y_b] \leftarrow \mathtt{COS}([1], [-1], [O])$. (4) SDT is used to transform $[y_b]$ into $[y_b]_{pk_{\mathscr{B}}}$, and sends the transformed value back to $\mathscr{B}$.

Step-3(@DT): Once $[y_b]_{pk_{\mathscr{B}}}$ is received, $\mathscr{B}$ uses his/her own private key $sk_{\mathscr{B}}$ to obtain the final result y_b.

6 Security Analysis

In this section, we will show that our POD can achieve its security level defined in Section 3.2.

6.1 Security of DT-PKC

The DT-PKC scheme is semantically secure in the semi-trusted model.

Theorem 1. *The **DT-PKC** scheme described in Section 2.2 is semantically secure, assuming the semantic security of the underlying Paillier cryptosystem.*

Proof. The security proof of the DT-PKC is presented in our previous work [12]. □

6.2 Security of Sub-Protocols

Here, we prove the security of our protocols for the specific scenario of its functionality, which involves three parties, namely: challenge DT/DP/AP (a.k.a. "$\mathscr{D}_M$"), CP (a.k.a. "S_1"), and CSP (a.k.a."S_2"). We construct three simulators $\mathsf{Sim} = (\mathsf{Sim}_{\mathscr{D}_M}, \mathsf{Sim}_{S_1}, \mathsf{Sim}_{S_2})$ for adversaries $(\mathcal{A}_{\mathscr{D}_M}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$ that corrupt $\mathscr{D}_M, S_1, S_2$, respectively. We refer the reader to [12], [18] for the general case definitions under non-colluding semi-honest adversaries.

Theorem 2. *SDT securely transforms the encryption domain in the presence of semi-honest (non-colluding) adversaries $\mathcal{A} = (\mathcal{A}_{\mathscr{D}_M}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.*

Proof. We only provide a proof to show how to construct three independent simulators $\mathsf{Sim}_{\mathscr{D}_M}, \mathsf{Sim}_{S_1}, \mathsf{Sim}_{S_2}$.

$\mathsf{Sim}_{\mathscr{D}_M}$ receives x as input and then simulates $\mathcal{A}_{\mathscr{D}_M}$ as follows: it generates encryption $[x]_{pk_M} \leftarrow \mathtt{Enc}(x, pk_M)$ of x under public key pk_M. Finally, it returns $[x]_{pk_M}$ and outputs $\mathcal{A}_{\mathscr{D}_M}$'s entire view. The view of $\mathcal{A}_{\mathscr{D}_M}$ consists of the encrypted data. The views of $\mathcal{A}_{\mathscr{D}_M}$ in the real and the ideal executions are indistinguishable due to the semantic security of DT-PKC.

Sim_{S_1} simulates $\mathcal{A}_{S_1}$ as follows: First, it generates (fictitious) encryption of the inputs $[\hat{x}]_{pk_\sigma}$ by running $\mathtt{Enc}(\cdot)$ on randomly chosen $\hat{x}$, randomly generates $\hat{r} \in \mathbb{Z}_N$ and computes $[\hat{r}]_{pk_\sigma}$, calculates $\hat{X}_1$, and then calculates $\hat{X}'_1$ using $\mathtt{PD1}(\cdot)$. Sim_{S_1} sends the encryption $\hat{X}_1$ and $\hat{X}'_1$ to $\mathcal{A}_{S_1}$. If $\mathcal{A}_{S_1}$ replies with $\perp$, then Sim_{S_1} returns $\perp$.

The view of $\mathcal{A}_{S_1}$ consists of the encrypted data it created. In both real and ideal executions, $\mathcal{A}_{S_1}$ receives the encryption $\hat{X}_1, \hat{X}'_1$. In the real world, it is guaranteed by the fact that the $\mathscr{D}_M$ (DT or DP or AP) is honest and the semantic security of DT-PKC. The views of $\mathcal{A}_{S_1}$ in the real and the ideal executions are indistinguishable.

Sim_{S_2} simulates $\mathcal{A}_{S_2}$ as follows: it randomly chooses $\hat{h}$, uses the $\mathtt{Enc}(\cdot)$ to obtain $[\hat{h}]_{pk_\sigma}$, and then sends the encryption to $\mathcal{A}_{S_2}$. If $\mathcal{A}_{S_2}$ replies with $\perp$, then Sim_{S_2} returns $\perp$.

The view of $\mathcal{A}_{S_2}$ consists of the encrypted data it created. In both real and ideal executions, $\mathcal{A}_{S_2}$ receives the output encryption $[\hat{h}]_{pk_\sigma}$. In the real world, it is guaranteed by the semantic security of DT-PKC. The views of $\mathcal{A}_{S_2}$ in the real and the ideal executions are indistinguishable. □

Theorem 3. *FApx securely performs approximation over ciphertext in the presence of semi-honest (non-colluding) adversaries* $\mathcal{A} = (\mathcal{A}_{\mathscr{D}_M}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.

Proof. The construction of three independent simulators $\mathsf{Sim}_{\mathscr{D}_M}, \mathsf{Sim}_{S_1}, \mathsf{Sim}_{S_2}$ in FApx is similar to those in Theorem 2 (see supplemental material, available online for the proof). □

The security proofs for (FAbs, SCOS, FCOS, MAX, TOP1) in Section 4.3.2 and SKer, SType1, SType2 in Section 5.2 are similar to those of FApx in the presence of semi-honest (non-colluding) adversaries $\mathcal{A} = (\mathcal{A}_{\mathscr{D}_M}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.

6.3 Security of POD Framework

The security of POD can be guaranteed by the following two theorems. Here, to prove the training phase of POD, we construct $n+2$ parties, challenge DP i (a.k.a. "$\mathscr{D}_i$", $i = 1, \ldots, n$), AP (a.k.a. "$\mathscr{A}$"), CP (a.k.a. "S_1"), and CSP (a.k.a."S_2"). We construct $n+3$ simulators $\mathsf{Sim} = (\mathsf{Sim}_{\mathscr{D}_i}, \mathsf{Sim}_{\mathscr{A}}, \mathsf{Sim}_{S_1}, \mathsf{Sim}_{S_2})$ for adversaries $(\mathcal{A}_{\mathscr{D}_i}, \mathcal{A}_{\mathscr{A}}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$ that corrupt $\mathscr{D}_i (i = 1, \ldots, n), \mathscr{A}, S_1, S_2$, respectively.

Theorem 4. *The POD framework securely achieves secure SVM training on ciphertext in the presence of semi-honest (non-colluding) adversaries* $\mathcal{A} = (\mathcal{A}_{\mathscr{D}_i}, \mathcal{A}_{\mathscr{A}}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.

Proof. The constructions of $\mathsf{Sim}_{\mathscr{D}_i}$ and $\mathsf{Sim}_{\mathscr{A}}$ for this theorem are analogous to $\mathsf{Sim}_{\mathscr{D}_M}$ for Theorem 2 while the constructions of Sim_{S_1} and Sim_{S_2} for this theorem are similar to the corresponding Sim_{S_1} and Sim_{S_2} in Theorem 3, respectively. The difference is that we need to use simulators instead of corresponding secure basic and enhanced protocols in the SVM training phase. □

To prove the decision phase of POD, we need additional parties DT (a.k.a. "$\mathscr{B}$"), and construct simulator $\mathsf{Sim}_{\mathscr{B}}$ in addition to $\mathsf{Sim}_{\mathscr{A}}, \mathsf{Sim}_{\mathscr{D}_i}, \mathsf{Sim}_{S_1}, \mathsf{Sim}_{S_2}$ $(i = 1, \ldots, n)$.

Theorem 5. *The POD framework securely achieves secure SVM classification in the presence of semi-honest (non-colluding) adversaries* $\mathcal{A} = (\mathcal{A}_{\mathscr{A}}, \mathcal{A}_{\mathscr{B}}, \mathcal{A}_{\mathscr{D}_i}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$.

Proof. The constructiona of these simulators are similar to those in the proof of Theorem 4. □

Here, we also give an analysis to demonstrate that our POD can resist system active adversary $\mathcal{A}_1^*$, $\mathcal{A}_2^*$ and $\mathcal{A}_3^*$ defined in Section 3.2. The analysis is described as follows: If $\mathcal{A}_1^*$, $\mathcal{A}_2^*$ and $\mathcal{A}_3^*$ eavesdrop on the transmission between the challenge DT and CP, on the transmission between the challenge DP and CP, and on the transmission between the AP and the CP, all data transmitted in these three links will be obtained by $\mathcal{A}_1^*$, $\mathcal{A}_2^*$ and $\mathcal{A}_3^*$, respectively. Moreover, the ciphertext results (obtained by executing basic secure fraction computation protocols and secure SVM training and classification) transmitted between CP and CSP may also be available to $\mathcal{A}_1^*$, $\mathcal{A}_2^*$ and $\mathcal{A}_3^*$ due to the eavesdropping. However, as these data are encrypted before transmission, $\mathcal{A}_1^*$ cannot decrypt the challenge DT's ciphertext without knowing the challenge DT's private key, $\mathcal{A}_2^*$ cannot decrypt the challenge DP's ciphertext without knowing the challenge DP's private key, and $\mathcal{A}_3^*$ cannot decrypt the challenge AP's ciphertext without knowing the challenge AP's private key. Thus, it can be guaranteed by the semantic security of the DT-PKC cryptosystem. Next, we assume that $\mathcal{A}_1^*$, $\mathcal{A}_2^*$ and/or $\mathcal{A}_3^*$ has/have compromised the CP to obtain the system strong partial private key. However, $\mathcal{A}_1^*$, $\mathcal{A}_2^*$ and $\mathcal{A}_3^*$ cannot recover the ciphertext, as the strong private key is randomly split by executing KeyS algorithm of DT-PKC. Even when the CSP is compromised, none of the three adversaries ($\mathcal{A}_1^*$, $\mathcal{A}_2^*$ and $\mathcal{A}_3^*$) can obtain useful information as our protocols use the known technique of "blinding" the plaintext [19]: given an encryption of a message, we use the additively homomorphic property of the DT-PKC cryptosystem to add a random message to the plaintext. Therefore, original plaintext is "blinded".

7 Performance Analysis

In this section, findings of the experiment and theoretical analysis of POD will be presented.

7.1 Experiment Analysis

7.1.1 Performance of Basic Secure Protocols

We evaluated the factors that impact the performance of the proposed protocols in our testbed with personal computer (PC) with a 3.6 GHz eight-cores processor and 12 GB RAM memory, and a customized simulator built using Java. Here, we assumed N to be 1,024 bits to achieve 80-bit security levels [31]. Next, we evaluated the runtime of basic crypto by executing each algorithm 1,000 times, and DT-PKC takes 14.509 ms to run Enc, 7.254 ms to run SDec, 7.379 ms to run WDec, 21.481 ms to run DP1, and 20.998 ms to perform DP2. For the secure protocols, there are three

TABLE 2
Performance of the Protocols ($\mu = 70$, $n = 5$, $t = 3$, $d = 2$)

	Computation Cost (s)							Communication Overhead (KB)						
N	512	768	1024	1280	1536	1792	2048	512	768	1024	1280	1536	1792	2048
SDT	0.01	0.04	0.11	0.20	0.36	0.55	0.78	1.12	1.68	2.25	2.81	3.37	3.93	4.50
FAdd	0.11	0.31	0.68	1.34	2.39	3.59	5.09	3.00	4.49	5.99	7.49	8.99	10.49	11.99
FSub	0.10	0.32	0.67	1.34	2.39	3.66	5.08	2.99	4.49	5.99	7.49	8.99	10.49	11.99
FMul	0.07	0.22	0.46	0.91	1.59	2.39	3.43	2.00	2.99	3.99	4.99	5.99	7.00	7.99
FCmp	0.09	0.27	0.58	1.10	2.08	3.02	4.28	2.62	3.93	5.24	6.55	7.87	9.18	10.49
FEQ	0.18	0.55	1.18	2.33	4.14	6.21	8.99	5.24	7.86	10.49	13.10	15.74	18.36	20.99
FIP	0.44	1.25	2.83	5.17	9.10	14.53	22.13	11.98	17.96	23.96	29.97	35.95	41.95	47.98
FApx	2.76	8.30	18.83	37.46	64.13	98.50	141.52	91.03	136.83	182.48	228.29	273.99	319.38	365.10
SKer	0.13	0.43	0.97	1.79	3.14	5.00	7.65	4.49	6.73	8.98	11.24	13.48	15.73	17.99
STyp1	1.37	4.22	9.62	17.96	31.12	49.24	75.38	42.37	63.58	84.89	106.11	127.36	148.59	169.83
STyp2	1.40	4.33	9.88	18.14	31.83	50.54	77.29	43.36	65.08	86.88	108.61	130.36	152.08	173.82
FAbs	0.06	0.19	0.40	0.77	1.38	2.14	3.14	1.62	2.43	3.25	4.06	4.87	5.68	6.49
SCOS	0.08	0.23	0.51	1.02	1.79	2.70	3.91	2.00	2.99	3.99	4.99	6.00	6.99	7.99
FCOS	0.16	0.47	1.04	2.02	3.60	5.40	7.82	3.99	5.99	7.99	9.98	11.99	13.99	15.98
MAX	0.70	2.16	4.71	9.20	16.40	24.66	35.71	18.60	27.88	37.20	46.49	55.84	65.14	74.45
TOP-1	2.80	8.62	19.09	36.91	65.44	98.46	142.65	55.78	83.62	111.59	139.46	167.51	195.43	223.34
SKKT	0.56	1.70	3.72	7.25	12.80	19.46	27.95	16.47	24.70	32.96	41.18	49.47	57.71	65.95
Ssel	8.29	23.38	55.17	108.41	192.42	288.53	417.81	213.15	311.57	426.41	532.87	640.08	746.77	853.41
SMMO	8.14	25.05	54.41	106.73	188.135	284.34	411.54	183.70	275.39	367.49	459.25	551.67	643.56	735.49
SVM	1.50	4.68	10.64	19.71	33.40	54.14	78.45	48.35	72.56	96.87	121.10	145.34	169.52	193.81

factors that impact the performance, namely: 1) the crypto (DT-PKC) parameter N, 2) the length of vector t, and 3) the number of instances n.

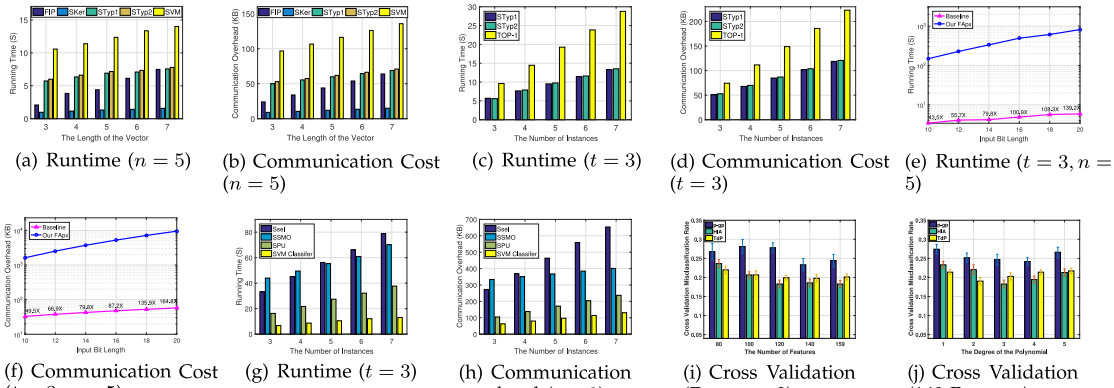
From Table 2, both computational cost and communication overhead of the secure protocol increase with N, as DT-PKC requires more time to process the encrypted data and storage to store the ciphertext. From Figs. 4a and 4b, both computational cost and communication overhead of `FIP`, `SKer`, `STyp1` and `STyp2` increase with t. This is because computing encrypted data requires computation and communication resources. From Figs. 4c and 4d, the computational cost and communication overhead of `SKer`, `STyp1` and `STyp2` increase with n, since more ciphertexts are calculated when n increases.

7.1.2 Performance of POD

Similar to the secure fraction protocols, there are also three factors that impact the performance of POD (including `Ssel`, `SSMO`, `SPU`, and `SVM` classifier), namely: 1) the DT-PKC parameter N, 2) the length of vector t, and 3) the number of instances n. From Table 2, we observed that both computational cost and communication overhead of these four components increase with N. It is because DT-PKC and all basic components of these four protocols in Section 2.2 increase with N. Furthermore, only the computational cost and communicational overhead of the `SVM` classifier increase with t (see Figs. 4a and 4b), as `SVM` classifier needs to use `STyp2` for computation. We observe that `Ssel`, `SSMO`, `SPU`, and `SVM` classifier increase with n as all these four protocols use all training dataset for computation (see Figs. 4g and 4h).

7.1.3 Real-World Dataset Experiment

As SVM is especially suitable for processing high-dimensional dataset, we use three real-world drug datasets from [20] that have been used for prediction of pharmacokinetic and toxicological properties of chemical agent (drug) such as p-glycoprotein substrate (p-gp, including substrates/nonsubstrates classes, a total of 201 instances), human intestinal absorption of molecules (HIA, including absorbable/nonabsorbable classes, a total of 196 instances), and agents that cause torsades de pointes (TdP, including TdP causing/non-causing classes, a total of 361 instances). These



(a) Runtime ($n = 5$) (b) Communication Cost ($n = 5$) (c) Runtime ($t = 3$) (d) Communication Cost ($t = 3$) (e) Runtime ($t = 3, n = 5$)

(f) Communication Cost ($t = 3, n = 5$) (g) Runtime ($t = 3$) (h) Communication overhead ($t = 3$) (i) Cross Validation (Degree = 3) (j) Cross Validation (160 Features)

Fig. 4. Evaluation findings.

three datasets require 159 descriptors (features) and one decision to describe.

To evaluate the effectiveness of POD, we use 10-fold cross-validation to determine misclassification rate influenced by the number of features (Fig. 4i) and the degree of polynomial in kernel function (Fig. 4j). From Fig. 4i, we observed that misclassification rate decreases when the number of features increase (best 10/6.0%/6.4 percent misclassification rate for p-gp/HIA/TdP), and Fig. 4j shows that the misclassification rate decreases when the polynomial degree increases (best 16.3/9.6/7.6 percent misclassification rate for p-gp/HIA/TdP). In the secure SVM training phase, it takes 3.05/2.95/5.49 hours (85.85/76.59/154.35 MB) to train the secure SVM in one round for p-gp/HIA/TdP respectively (include 0.68/0.64/1.25/ hours (18.7/18.2/33.62 MB) to run `Ssel`, 2.07/2.02/3.7 hours (59.5/58.1/107.0 MB) to run `SMMO`, and 0.3/0.29/0.54 hours (7.65/7.46/13.73 MB) to run `SPU`). For new instance SVM classification, it takes 1.5/1.4/2.8 hours (49.5/48.3/88.7 MB) for p-gp/HIA/TdP dataset, respectively.

7.2 Theoretical Analysis

7.2.1 Computational Overhead

Let us assume that one regular exponentiation operation with an exponent of length $\|N\|$ requires $1.5\|N\|$ multiplications [21] (e.g., the length of r is $\|N\|$, and computation of g^r requires $1.5\|N\|$ multiplications, denoted as $1.5\|N\|$ muls). As exponentiation operation is significantly more costly than the addition and multiplication operations, we ignored the fixed numbers of addition and multiplication operations in our analysis. As basic `SM` and `SLT` need $45\|N\|$ muls and $30\|N\|$ muls respectively, `FAdd` requires $135\|N\|$ muls, while `FSub` requires $138\|N\|$ muls to process. Similarly, it costs $90\|N\|$ muls to run `FMul` and `FDiv`. Also, it costs $120\|N\|$ muls to run `FCmp`, $240\|N\|$ muls to run `FEqu`, $93\|N\|$ muls to run `COS`, $78\|N\|$ muls to run `FAbs`, $186\|N\|$ muls to run `FOS`, $864\|N\|$ muls to run `MAX`, $\mathcal{O}(\delta\|N\|)$ muls to run `FIP`, and $\mathcal{O}(\lfloor\log_2\delta\rfloor\|N\|)$ muls to run `TOP1`, where δ is the number of the input encryption. For the enhanced protocols, `SKer` needs $\mathcal{O}((t+\lfloor\log_2 d\rfloor)\|N\|)$ muls, both `STyp1` and `STyp2` need $\mathcal{O}((t+\lfloor\log_2 d\rfloor)n\|N\|)$ muls, where d is the degree of the polynomial kernel. For the POD, it takes $\mathcal{O}(n\|N\|)$ muls to run `Ssel`, $\mathcal{O}(n^2\|N\|)$ muls to run `SMMO`, and $\mathcal{O}(n^2\|N\|)$ muls to run `SPU` for one round respectively (total $\mathcal{O}(n^2\|N\|)$ muls for secure SVM training phase). Finally, it takes $\mathcal{O}(n\|N\|)$ muls for CP & CSP for training completion check, and $\mathcal{O}((t+\lfloor\log_2 d\rfloor)n\|N\|)$ muls to run the `SVM` classifier.

7.2.2 Communication Overhead

In the DT-PKC scheme, $[x]$ needs $4\|N\|$ bits and partially decrypted ciphertext need $2\|N\|$ bits to transmit. For the basic protocol, both `FAdd` and `FSub` costs $48\|N\|$ bits to process. Similarly, it costs $32\|N\|$ bits to run `FMul` and `FDiv`. Also, it costs $42\|N\|$ bits to run `FCmp`, $84\|N\|$ bits to run `FEqu`, $26\|N\|$ bits to `FAbs`, $32\|N\|$ bits to run `COS`, $64\|N\|$ bits to run `FOS`, $298|N|$ bits to run `MAX`, $\mathcal{O}(\delta\|N\|)$ bits to run `FIP`, and $\mathcal{O}(\lfloor\log_2\delta\rfloor\|N\|)$ bits to run `TOP1`, where δ is the number of the input encryption. For the enhanced protocols, `SKer` needs $\mathcal{O}((t+\lfloor\log_2 d\rfloor)\|N\|)$ bits, and both `STyp1` and `STyp2` need $\mathcal{O}((t+\lfloor\log_2 d\rfloor)n\|N\|)$ bits. For the POD, it takes $\mathcal{O}(n\|N\|)$ bits to run `Ssel`, $\mathcal{O}(n^2\|N\|)$ bits to run `SMMO`, and $\mathcal{O}(n^2\|N\|)$ bits to run `SPU` for one round respectively. Finally, it takes $\mathcal{O}(n\|N\|)$ bits for CP & CSP for training completion check, and $\mathcal{O}((t+\lfloor\log_2 d\rfloor)n\|N\|)$ bits to run the `SVM` classifier.

7.3 Comparative Analysis

Here, we present the comparative summary of the proposed POD and existing privacy-preserving SVM classifier (PP-SVM) and secure data approximate method. The storage of the existing PP-SVM can be categorized under two types, namely: secure distributed storage and secure centralized storage. Secure distributed storage PP-SVM uses secret sharing to split the user's data into different partitions, which can be categorized into vertically partitioned [22], horizontally partitioned [23], and arbitrarily partitioned [24]. However, this strategy has two main drawbacks. First, it stores the shares of one instance across different servers. Second, all computations need to be online at the same time when performing the secure computation. To overcome the limitations, PP-SVM with centralized storage schemes [25], [26] have been proposed in the literature. Such schemes allow the data owner to encrypt and outsource the data in the single storage server. Although the centralized strategy can achieve privacy-preserving classification on-the-fly, it requires multiple communication rounds between the data owner and the storage server. Lin and Chen [27] presented a solution to address the privacy concerns during the outsourced SVM training, but its security is too weak for drug discovery. To solve the plaintext overflow problem, we proposed a secure data approximate method to securely deduce the plaintext length in a previous work [13]. However, it is inefficient due to the use of the time consuming secure division protocol as the basis. The runtime of our new secure fraction approximation protocol is much faster than the corresponding one in [13] (139.2 times faster than [13] when the plaintext bit length is 20, also see Figs. 4e and 4f for comparison). A comprehensive comparison between our POD and the above schemes is presented in Table 3.

7.4 Alternate and Further Optimization Methods

In addition to secure fraction computation, it appears that all data can be directly stored as the fixed-point number format, and expanded to an integer. Then, secure computation can directly use secure integer protocols to construct our POD. Compared with secure fraction storage, it only requires $2\|N\|$ bits to store instead of $4\|N\|$ bits, where N is the maximum value of crypto plaintext domain. However, our POD needs to perform division calculation, and the secure integer division protocol requires $\mathcal{O}(\mu^2\|N\|+\mu^3)$ muls, where μ is the bit-length of the integer in the system. The latter is impractical for large plaintext (a.k.a μ is large), while our secure fraction computation only requires $\mathcal{O}(\|N\|)$ muls. Another alternative solution is to use floating-point format to securely store information, which allows one to achieve high precision computation. However, we would need to encrypt the sign, normalized significand, and exponent. This requires $6\|N\|$ bits for a single number. Moreover, the secure floating-point addition requires $\mathcal{O}(\eta\|N\|)$ muls, while our secure fraction computation only needs $\mathcal{O}(1)$ muls (i.e., 4 muls), where η denotes the bit-length of significand.

TABLE 3
A Comparative Summary

Function/Algorithm	[22]	[23]	[24]	[25]	[26]	[27]	[13]	Proposed
Support Multi-User	✓	✓	✓	×	×	×	×	✓
Single Server Storage	×	×	×	✓	✓	✓	✓	✓
Solve Data Synchronization Problem	×	×	×	✓	✓	✓	✓	✓
Communication Round (User & Server)	One	One	One	Multiple	Multiple	Multiple	Multiple	One
Solve Plaintext Overflow	×	×	×	×	×	×	✓	✓
Support SVM training	×	×	×	×	×	✓	N.A.	✓
Security Level	Weak	Weak	Weak	Weak	Weak	Weak	Strong	Strong
Semi-honest Model	✓	✓	✓	✓	✓	✓	✓	✓

The computation cost for secure training phase is still high for real-world deployment. Thus, we give two optimization methods to further decrease the outsource computation time: 1) Multi-threaded programming. In a worst-case scenario, we only use a single-threaded program in our evaluation, and the multi-threaded programming could achieve a significantly lower runtime for POD; 2) High-performance hardware. We only use common PC to simulate the outsourced secure computation. In a real-world deployment, the secure protocols would most probably be deployed on machines with GPUs.

8 Related Work

Searching over billions of molecules to find possible drug formula, drug design using machine learning is what some may consider the "Holy Grail". SVM classifier is one of the most robust and highly accurate intelligent classification techniques in machine learning for designing drug applications. For example, Burbidge et al. [9] demonstrated that using SVM classification to predict the inhibition of dihydrofolate reductase by pyrimidines, has a higher classification rate compared to other machine learning techniques. There have been other attempts to use SVM in the drug discovery/designing process. For example, Byvatov et al. [28] used SVM for drug/nondrug classification in the early phase virtual compound filtering and screening, and Tou et al. [29] used SVM as a basis to achieve virtual screening and ligand-based drug design for Traditional Chinese Medicine (TCM). A number of machine learning techniques for drug discovery have also been proposed in the literature, such as extreme learning machine [30], deep learning [31], and naïve Bayes classifer [32].

To protect the privacy of SVM classifier, Yu et al. [22] presented a scalable solution for privacy-preserving SVM classification on vertically partitioned data. As the scheme uses linear kernel function, Yu et al. [23] constructed a privacy-preserving SVM on horizontally partitioned data with (Gaussian) radial basis function kernel. To support arbitrarily partitioned data, Vaidya et al. [24] constructed a privacy-preserving SVM, and quantified the security and efficiency of the method. These schemes use multiple servers to store data, which leads to data synchronization problem across different storage servers. To solve the problem, Rahulamathavan et al. [25] constructed a privacy-preserving clinical decision support system using the SVM classifier. Rahulamathavan et al. [26] also built a privacy-preserving outsourced multi-class SVM to solve the multi-class problems on-the-fly. Clients in both schemes need to encrypt their test sample prior to outsourcing to the cloud. However, multiple communication rounds between the client and the cloud are needed in both schemes of [25] and [26]. To minimize the communication rounds for clients, other frameworks have also been proposed in the literature. However, none of these schemes support secure SVM training (an important phase in drug discovery). Lin and Chen [27] perturbed the data using random linear transformation to construct the privacy-preserving outsourcing of the SVM training. Unfortunately, it only supports single data owner. Later, Lin and Chen [33] considered the privacy problem of the SVM training dataset and send the encrypted SVM to a client for classification. Note that both schemes in [27] and [33] are not semantically secure (See the security model in [34]).

9 Conclusion

In this paper, we proposed POD, a new privacy-preserving outsourced drug discovery in the cloud. POD is designed to facilitate drug manufacturers to securely outsource their formulas to the cloud for storage and SVM training. The trained SVM model could be used for authorized client's compound classification in a privacy-preserving way. Specifically, we designed a secure domain transformation protocol and several basic secure computation components for secure outsourced computation across different parties. We also built two key secure components (i.e., secure parameter selection and secure sequential minimal optimization) to achieve privacy-preserving SVM training in drug discovery. In the future, we will be extending our approach to support more sophisticated data mining method in order to support very large dataset in drug discovery.

Acknowledgments

This research is supported in part by the AXA Research Fund, National Natural Science Foundation of China under Grant No. 61702105 and No. 61402112.

References

[1] J. P. Hughes, S. Rees, S. B. Kalindjian, and K. L. Philpott, "Principles of early drug discovery," *Brit. J. Pharmacology*, vol. 162, no. 6, pp. 1239–1249, 2011.

[2] The price of health: The cost of developing new medicines. (2016). [Online]. Available: https://www.theguardian.com/healthcare-network/2016/mar/30/new-drugs-development-costs-pharma

[3] I. Khanna, "Drug discovery in pharmaceutical industry: Productivity challenges and trends," *Drug Discovery Today*, vol. 17, no. 19, pp. 1088–1102, 2012.

[4] M. A. Lill and M. L. Danielson, "Computer-aided drug design platform using PyMOL," *J. Comput.-Aided Molecular Des.*, vol. 25, no. 1, pp. 13–19, 2011.

[5] Research and markets, global drug discovery technologies market analysis & trends - industry forecast to 2025. (2017). [Online]. Available: http://www.researchandmarkets.com/research/n5klng/global_drug
[6] Y. Zhang and J. C. Rajapakse, *Machine Learning in Bioinformatics*. Hoboken, NJ, USA: John Wiley & Sons, 2009, vol. 4.
[7] J. B. Mitchell, "Machine learning methods in chemoinformatics," *Wiley Interdisciplinary Rev.: Comput. Molecular Sci.*, vol. 4, no. 5, pp. 468–481, 2014.
[8] T. Joachims, "Making large-scale SVM learning practical," *Advances Kernel Methods-Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, MIT Press Cambridge, MA, USA, pp. 169–184, 1999.
[9] R. Burbidge, M. Trotter, B. Buxton, and S. Holden, "Drug design by machine learning: Support vector machines for pharmaceutical data analysis," *Comput. Chemistry*, vol. 26, no. 1, pp. 5–14, 2001.
[10] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," *Crypto ePrint Archive*, 2014, https://eprint.iacr.org/2014/331.pdf
[11] G. Cano, et al., "Automatic selection of molecular descriptors using random forest: Application to drug discovery," *Expert Syst. Appl.*, vol. 72, pp. 151–159, 2017.
[12] X. Liu, R. Lu, J. Ma, L. Chen, and B. Qin, "Privacy-preserving patient-centric clinical decision support system on naive bayesian classification," *IEEE J. Biomed. Health Informat.*, vol. 20, pp. 655–668, 2016.
[13] X. Liu, R. Choo, R. Deng, R. Lu, and J. Weng, "Efficient and privacy-preserving outsourced calculation of rational numbers," *IEEE Trans. Dependable and Secure Comput.*, vol. 15, no. 1, pp. 27–39, 2018.
[14] B. K. Samanthula, H. Chun, and W. Jiang, "An efficient and probabilistic secure bit-decomposition," in *Proc. 8th ACM SIGSAC Symp. Inform. Comput. Commun. Security*, 2013, pp. 541–546.
[15] X. Liu, R. H. Deng, K.-K. R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," *IEEE Trans. Inform. Forensics Security*, vol. 11, no. 11, pp. 2401–2414, Nov. 2016.
[16] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," *Advances in kernel methods*, MIT Press Cambridge, MA, USA, pp. 185–208, 1999.
[17] R. Todeschini and V. Consonni, *Handbook of Molecular Descriptors*. Hoboken, NJ, USA: John Wiley & Sons, 2008, vol. 11.
[18] S. Kamara, P. Mohassel, and M. Raykova, "Outsourcing multi-party computation," *IACR Cryptology ePrint Archive*, vol. 2011, 2011, Art. no. 272. [Online]. Available: [Online]. Available: http://eprint.iacr.org/2011/272
[19] A. Peter, E. Tews, and S. Katzenbeisser, "Efficiently outsourcing multiparty computation under multiple keys," *IEEE Trans. Inform. Forensics Security*, vol. 8, no. 12, pp. 2046–2058, Dec. 2013.
[20] Y. Xue, Z.-R. Li, C. W. Yap, L. Z. Sun, X. Chen, and Y. Z. Chen, "Effect of molecular descriptor feature selection in support vector machine classification of pharmacokinetic and toxicological properties of chemical agents," *J. Chemical Inform. Comput. Sci.*, vol. 44, no. 5, pp. 1630–1638, 2004.
[21] D. E. Knuth, "The art of computer programming, volume 2 (3rd ed.): Seminumerical algorithms," Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1997.
[22] H. Yu, J. Vaidya, and X. Jiang, "Privacy-preserving SVM classification on vertically partitioned data," in *Proc. 10th Pacific-Asia Conf. Adv. Knowl. Discovery Data Mining*, 2006, pp. 647–656.
[23] 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.
[24] J. Vaidya, H. Yu, and X. Jiang, "Privacy-preserving SVM classification," *Knowl. Inf. Syst.*, vol. 14, no. 2, pp. 161–178, 2008.
[25] Y. Rahulamathavan, S. Veluru, R. C.-W. Phan, J. A. 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.
[26] Y. Rahulamathavan, R. C.-W. Phan, S. Veluru, K. Cumanan, and M. Rajarajan, "Privacy-preserving multi-class support vector machine for outsourcing the data classification in cloud," *IEEE Trans. Dependable Secure Comput.*, vol. 11, no. 5, pp. 467–479, May 2014.
[27] K. Lin and M. Chen, "Privacy-preserving outsourcing support vector machines with random transformation," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 363–372.
[28] E. Byvatov, U. Fechner, J. Sadowski, and G. Schneider, "Comparison of support vector machine and artificial neural network systems for drug/nondrug classification," *J. Chemical Inform. Comput. Sci.*, vol. 43, no. 6, pp. 1882–1889, 2003.
[29] W. I. Tou, S.-S. Chang, C.-C. Lee, and C. Y.-C. Chen, "Drug design for neuropathic pain regulation from traditional chinese medicine," *Scientific Reports*, vol. 3, 2013, Art. no. 844.
[30] W. M. Czarnecki, "Weighted tanimoto extreme learning machine with case study in drug discovery," *IEEE Comput. Intell. Mag.*, vol. 10, no. 3, pp. 19–29, Mar. 2015.
[31] E. Gawehn, J. A. Hiss, and G. Schneider, "Deep learning in drug discovery," *Molecular Informat.*, vol. 35, no. 1, pp. 3–14, 2016.
[32] A. Bender, H. Y. Mussa, R. C. Glen, and S. Reiling, "Molecular similarity searching using atom environments, information-based feature selection, and a naive Bayesian classifier," *J. Chemical Inform. Comput. Sci.*, vol. 44, no. 1, pp. 170–178, 2004.
[33] K.-P. Lin and M.-S. Chen, "Privacy-preserving outsourcing support vector machines with random transformation," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 363–372.
[34] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge, U.K.: Cambridge University Press, vol. 2, 2009.

Ximeng Liu (S'13-M'16) received the BSc degree in electronic engineering from Xidian University, Xi'an, China, in 2010, and the PhD degree in cryptography from Xidian University, China, in 2015. Now, he is a research fellow in the School of Information System, Singapore Management University, Singapore. He has published more than 80 research articles include the *IEEE Transactions on Information Forensics and Security*, the *IEEE Transactions on Dependable and Secure Computing*, the *IEEE Transactions on Computers*, the *IEEE Transactions on Industrial Informatics* the *IEEE Transactions on Services Computing* and the *IEEE Transactions on Cloud Computing*. His research interests include cloud security, applied cryptography and big data security. He is a member of the IEEE.

Robert H. Deng (F'16) is AXA chair professor of cybersecurity and professor of information systems in the School of Information Systems, Singapore Management University, since 2004. His research interests include data security and privacy, multimedia security, network and system security. He served/is serving on the editorial boards of many international journals, including the *IEEE Transactions on Information Forensics and Security*, and the *IEEE Transactions on Dependable and Secure Computing*. He is a fellow of the IEEE.

Kim-Kwang Raymond Choo (SM'16) received the PhD degree in information security from Queensland University of Technology, Australia, in 2006. He currently holds the Cloud Technology Endowed professorship with the University of Texas at San Antonio. He is the recipient of various awards including ESORICS 2015 Best Paper Award, Winning Team of the Germany's University of Erlangen-Nuremberg (FAU) Digital Forensics Research Challenge 2015, and British Computer Society's Wilkes Award in 2008. He is a fellow of the Australian Computer Society. He is a senior member of the IEEE.

Yang Yang (M'17) received the BSc degree from Xidian University, Xi'an, China, in 2006, and the PhD degree from Xidian University, China, in 2012. She was a research fellow (postdoctor) under supervisor Robert H. Deng in the School of Information System, Singapore Management University. Now, she is an associate professor in the college of mathematics and computer science, Fuzhou University. Her research interests include the area of information security and privacy protection. She is a member of the IEEE.