Textile & Leather Review

ISSN 2623-6281| www.tlr-journal.com | 10.31881/TLR

Cloud-Assisted Verifiable Support Vector Machine Training for IoT Devices

Shimao Yu

How to cite: Yu S. Cloud-Assisted Verifiable Support Vector Machine Training for IoT Devices. Textile & Leather Review. 2026; 9:4247-4277. https://doi.org/10.31881/TLR.2026.4247

How to link https://doi.org/10.31881/TLR.2026.4247

Published: 25 April 2026

Cloud-Assisted Verifiable Support Vector Machine Training for IoT Devices

Shimao Yu
College of Computer Science and Technology, Qingdao University, Qingdao, 266071, China.
yushimao@qdu.edu.cn

Article
https://doi.org/10.31881/TLR.2026.4247
Published 25 April 2026

ABSTRACT

Support vector machines (SVMs) are widely used for online inference in AI enabled cyber-physical systems (CPS) because of their strong generalization ability and effectiveness on high-dimensional, nonlinear data. To reduce deployment costs on edge and end devices and improve service availability, SVM inference is often outsourced to the cloud. However, if the cloud server is semi- honest or malicious, the messages exchanged during inference may leak sensitive information about both user inputs and model parameters. In addition, a malicious server may return incorrect or forged inference results, which can mislead downstream decisions and threaten system security. It is therefore important to develop SVM inference mechanisms for untrusted cloud environments that provide privacy, verifiability, and efficiency. To this end, we propose two privacy- preserving inference schemes for adversaries with different capabilities. For the semi-honest setting, we first propose OPVSVM, an efficient privacy-preserving inference scheme based on secret sharing. OPVSVM protects user data and intermediate values and reduces computation and communication overhead by optimizing key inference operators. For malicious cloud servers, we further propose WPVSVM, a verifiable privacy-preserving inference scheme that sup- ports result verification without revealing sensitive information. Experimental results show that, compared with representative baselines, both OPVSVM and WPVSVM improve runtime efficiency and demonstrate practical performance for secure cloud-based SVM inference in AI-enabled CPS.

KEYWORDS
privacy-preserving, support vector machine, cloud computing, cyber-physical systems, IoT devices

INTRODUCTION

Cyber-physical systems (CPS) are becoming increasingly interconnected, data- intensive, and autonomous. In domains such as industrial control, smart healthcare, intelligent transportation, and critical infrastructure, CPS rely heavily on online inference for perception, state estimation, and decision-making. Among classical machine learning methods, support vector machines (SVMs) remain attractive because of

their strong generalization ability and effectiveness on high-dimensional, nonlinear data. With the development of cloud computing infrastructures [1,2], SVM-based inference is increasingly deployed as a cloud service for CPS applications, including medical decision support, fault diagnosis, traffic event detection, and financial risk assessment. Particularly under the Machine-Learning-as-a-Service (MLaaS) paradigm, a practical commercial separation often exists between the entity owning the model and the entity providing the computation. Model Providers (MPs), such as healthcare institutions or AI startups, possess proprietary models and domain expertise but often lack the massively scalable infrastructure required to serve continuous IoT requests. Consequently, MPs outsource the heavy inference computation to commercial public Cloud Servers (CSs). Such deployment enables edge and end devices to access online inference capabilities without maintaining costly local computing platforms, thereby reducing operational cost and improving service availability for delay-tolerant CPS applications.

However, outsourcing inference to the cloud also introduces serious privacy and security concerns. Because CPS operate in a closed loop, cloud inference results may directly affect downstream physical actions. Once an inference service is monitored or compromised by an untrusted cloud, the impact can extend beyond data confidentiality to the physical world, potentially causing incorrect control actions, misdiagnosis, inappropriate treatment, or failures in critical infrastructure. In practical threat models, the cloud server is typically assumed to be either semi-honest or malicious. A semi-honest server follows the prescribed protocol but attempts to infer sensitive information from intermediate messages, where as a malicious server may deviate from the protocol by tampering with inputs, manipulating the computation, or forging outputs. Under such settings, the inference process may leak both model parameters and user data, leading to privacy breaches, model theft, and incorrect inference outcomes. Furthermore, in the MLaaS paradigm, the MP and the CS are separate entities with mutually distrustful interests: the MP wishes to protect its proprietary model parameters from the curious cloud, while the users wish to protect their private input data. Therefore, secure cloud-based SVM inference for AI-enabled CPS should provide privacy protection, result integrity, and practical efficiency under real-time and resource constraints.

Multiple privacy-preserving cloud computing frameworks for SVM have been pro- posed to date [3-9]. However, most existing solutions suffer from computational inefficiency and lack mechanisms for verifying

inference results. The few schemes that have designed a verification mechanism are also inadequate in their implementation. The schemes in [3-6] have low efficiency and lack verification mechanisms to detect malicious cloud servers. [7,8] designed verification for inference results using different methods. However, both may leak private data (e.g., machine learning model parameters [7] or users' private information [8]), thereby endangering data privacy. Although [9] has largely resolved the issues in previous work by improv- ing efficiency and proposing a secure verification mechanism, it still has significant room for improvement.

To address the issue of low computational efficiency, we devise a secure out- sourced SVM inference scheme that significantly enhances computational efficiency. Furthermore, to mitigate potential threats from malicious cloud servers, we propose a cloud-assisted privacy-preserving verification mechanism. The contributions of this work are summarized as follows:

• Problem formulation and threat model for AI-enabled CPS inference. We study privacy-preserving and trustworthy outsourced inference for SVM-based online decision making in AI-enabled CPS, where cloud inference results can directly affect downstream physical actions. We formalize two practical adversary models: a semi- honest cloud that follows the protocol but attempts to infer sensitive information from intermediate messages, and a malicious cloud that may deviate from the protocol by tampering with inputs or forging outputs.

• Efficient privacy-preserving SVM inference under the semi-honest model *OPVSVM*. We propose OPVSVM, an efficient outsourced SVM inference scheme built upon secret sharing. By protecting user inputs and intermediate values and optimizing key operators in the inference pipeline, OPVSVM significantly reduces computation and communication overhead while preserving confidentiality.

• Verifiable privacy-preserving SVM inference under the malicious model *WPVSVM* and evaluation. We further propose *WPVSVM*, which augments privacy-preserving inference with a verification mechanism to detect incorrect or forged inference results without revealing sensitive information. Extensive experiments on MNIST demonstrate that *OPVSVM* and *WPVSVM* achieve substantial performance gains over representative baselines, making them practical building blocks for secure and trustworthy cloud inference in AI-enabled CPS.

The paper is structured as follows. Section II introduces preliminaries, followed by the system and threat models in Section III. Section IV details the core security protocols. Section V describes the detailed implementation of our inference model. Section VI provides security analysis with formal proofs. Section VII discusses the experimental results. Section VIII surveys related work, and Section IX concludes.

PRELIMINARIES

Notation

We summaries the notation used in this paper in Table 1.

Support Vector Machine

SVM, one of the classical machine learning algorithms, is widely used for inference tasks. Given a training dataset with *H* samples, each sample consists of an *n*-dimensional feature vector $x_{i,j}$ and a corresponding class label $y_{i,j}$. The core idea of SVM is to classify data by finding an optimal decision boundary (hyper-plane) that best separates the training samples, with the samples closest to this hyperplane referred to as support vectors. For cases where the training data is linearly separable, inference can be achieved by identifying a linear decision boundary. However, when the data is not linearly separable, a kernel function $K(x_{i,j}, t)$ can be employed to map the data into a higher-dimensional space where it becomes linearly separable. Common kernel functions include the exponential kernel and Gaussian kernel, with the Gaussian kernel being particularly versatile. Therefore, this paper utilizes the Gaussian kernel for its broad applicability.

Table 1. Notations used in this paper

Symbols	Descriptions
MP	The Model Provider
U	User
CS_i	Cloud Server of ith
m	Number of classes

X_i	The index collection of support vectors of ith decision function
[[H]]	Share of two-party additive secret sharing
<H>	Share of three-party replication secret sharing
$[[H]]^B, < H >^B$	Share under binary sharing
$[[H]]_i, < H >_i$	The secret sharing shares of the ith user
λ	A scaling factor
$x_{i,j}$	The jth support vector of the ith class
b_i	The ith type of intercept term
L_i	The label of class i

To address multiclass problems, SVM transforms them into multiple binary inference tasks. There are two main approaches for this: One-versus-Rest (OVR) and One-versus-One (OVO). The OVR approach requires training *k* binary classifiers, whereas OVO requires training $k(k\_1)/2$ classifiers. For efficiency, we use the OVR method.

The SVM formulation can be expressed as follows:

$$ML = \max_{i=1,\ldots,m} \sum_{j \in X_i} \alpha_{i,j} y_{i,j} e^{-\lambda \left\| x_{i,j} - t \right\|_2^2} + b_i \quad (1)$$

where X_i represents the set of support vector indices for the *i*-th class. $x_{i,j}$ and $y_{i,j}$ denote the support vectors and their respective class labels, while $\alpha_{i,j}$ represents the Lagrange multiplier, and b_i is the intercept term. The vector *t* is an *n*-dimensional feature vector representing the input data to be classified. Let $C_{i,j} = a_{i,j}\, y_{i,j}$. Finally, ML indicates the class label corresponding to the maximum value of the decision

function where $K(x_{i,j}, t) = e^{-\lambda \left\| x_i - t \right\|_2^2}$.

Secret Sharing

Secret sharing is a technique that divides a secret into multiple shares and distributes these shares to different users, where only with a sufficient number of shares can the secret be reconstructed. The security of secret sharing primarily relies on mathematical principles, such as polynomial interpolation.

Two-party Addition Secret Sharing

In this paper, $[[x]]$ denotes additive secret sharing over the ring Z_{2^k}, where $x \in Z_{2^k}$. Operations with $k = 1$ are termed binary circuits, while those with larger k are called arithmetic circuits. The secret s is shared between two participants by splitting it into two shares s_1 and s_2. To reconstruct s, we compute $s \equiv s_1 + s_2 \pmod{p}$.

When computing the sum $x + y$ under secret sharing, the operation can be per- formed locally by participants through $[[x+y]]_i \equiv [[x]]_i + [[y]]_i$, followed by reconstructing $[[x + y]]_i$. This method also applies to subtraction operations.

To perform multiplicative operations, we utilize multiplication triples. A multiplication triple (a, b, c), where $c = a \cdot b$, is precomputed during the preprocessing phase. Each participant holds shares of the triple, denoted as $([[a]]_i,[[b]]_i,[[c]]_i)$. When computing $z = x \cdot y$, the following transformation is applied:

$$z = x + y = (x - a + a)(y - b + b) = (e + a)(f + b) = ef + eb + fa + ab \quad (2)$$

where $e = x - a$ and $f = y - b$. All participants locally compute $[[e]]_i$ and $[[f]]_i$, then reconstruct and publicly reveal e and f. Subsequently, $[[z]]_i$ can be locally computed using the aforementioned formula.

Three-party Replicated Secret Sharing

In this work, we employ a 2-out-of-3 replicated secret sharing scheme. The secret value x is shared among three cloud servers CS_0, CS_1, CS_2. Specifically, the ith participant CS_i receives a pair of random values $(x_{i-1(mod3)}, x_{i+1(mod3)})$, such that $x \equiv x_0 + x_1 + x_2 \pmod{M}$. This sharing is denoted as $\langle x \rangle$ throughout the paper.

For shared values $\langle x \rangle$ and $\langle y \rangle$, linear operations can be performed non-interactively. The addition of $\langle x \rangle$ and $\langle y \rangle$ is computed as $\langle x \rangle + \langle y \rangle := (x1 + y1, x2 + y2, x3 + y3)$. Subtraction follows an analogous procedure.

The multiplication between two shares is relatively complex. We can learn:

$$xy = (x_0 + x_1 + x_2)(y_0 + y_1 + y_2) \tag{3}$$

Let $z = x\,y = z0 + z1 + z2$ and $z0 = x1 \cdot y1 + x1 \cdot y2 + x2 \cdot y1$, $z1 = x2 \cdot y2 + x2 \cdot y0 + x0 \cdot y2$, $z2 = x0 \cdot y0 + x0 \cdot y1 + x1 \cdot y0$. As observed, each participant can only obtain one of these additive terms, where z_i is locally computed by CS_i. We execute a re-sharing protocol to enable participants to acquire another term. First, CS_i and CS_{i+1} pre-share a PRG key to generate identical random values $r_{i,i+1}$. A 3-out-of-3 zero-value sharing $(\alpha 0, \alpha 1, \alpha 2)$ is then constructed, where $\alpha_i = r_{i,i+1} \_\_ r_{i-1,i}$. Finally, $z'_i = z_i + \alpha_i$ is computed by CS_i and sent to CS_{i+1} to create the 2-out-of-3 share $((x \cdot y)i\text{-}1, (x \cdot y)i\text{+}1) = (z'_i, z'_i\text{-}1)$.

SYSTEM MODEL

This section outlines the system and threat models of the two distinct schemes. In standard secret-sharing literature, it is a common premise to assume that the participating computing servers do not collude. However, relying on multiple virtual instances hosted by a single commercial Cloud Service Provider (CSP) renders this assumption vulnerable. A compromised hypervisor or a malicious insider with global administrative privileges could potentially reconstruct the secret data, creating a single point of failure. To practically enforce the non-collusion assumption in a real-world deployment, our framework is designed to operate over a Multi-Cloud Architecture (or Cross-Cloud architecture). Rather than centralizing computation within one provider, the three computing servers are distributed across distinct, independently operated, and commercially competing CSPs.

While the broader domain of Cyber-Physical Systems includes high-speed, hard real-time applications (e.g., autonomous driving, high-frequency control loops), the cryptographic overhead inherent in secret-sharing-

based secure inference makes it unsuitable for sub-millisecond physical responses. Therefore, our proposed framework specifically targets soft real-time and delay-tolerant AI-enabled CPS applications. Typical scenarios include:

Smart Healthcare (Medical Decision Support): Analyzing patient data for disease diagnosis where a delay of several seconds is clinically acceptable and unnoticeable compared to human diagnostic time.

Industrial Internet of Things (Predictive Maintenance): Periodically evaluating sensor logs to predict equipment wear and tear, where analysis occurs in batches or intervals of seconds to minutes.

Smart Grid (Energy Consumption Analysis): Assessing grid health and user consumption patterns, which typically tolerate latency in the magnitude of seconds.

In these scenarios, the critical priority is the absolute privacy of user data and model parameters, for which a trade-off yielding a few seconds of latency is highly practical.

Efficient Privacy-Preserving Inference

Figure 1 illustrates the system model for our efficient privacy-preserving inference model. The model comprises three entities: The Model Provider (MP), Cloud Servers (CS), and User (U).

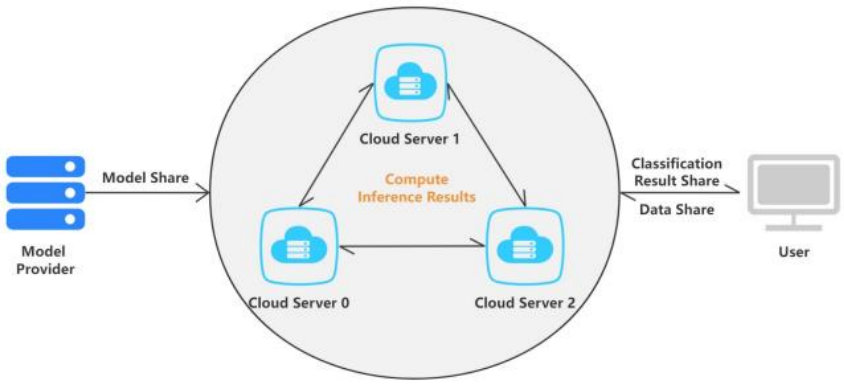

Figure 1. System Model for Efficient Privacy-Preserving Inference Framework

1. *The Model Provider:* MP maintains a pre-trained multi-class SVM classifier and outsources it to CS, enabling CS to provide secure inference services for U.

2. *User:* U takes the samples requiring inference as input and securely obtains the inference results by submitting computation requests to CS.

3. *Cloud Servers:* Three non-colluding CSs securely execute the SVM algorithm on the input samples provided by U, generate shares of the inference result, and distribute them back to U.

The detailed workflow of the inference model is structured as follows:

First, MP protects each model parameter by generating 2-out-of-3 replicated secret-sharing shares, distributing these shares to CS respectively. This enables CS to collaboratively utilize the SVM model for inference tasks. Subsequently, U generates 2-out-of-3 replicated secret-sharing shares of the to-be-classified sample and transmits them to CS. CS then jointly execute the SVM algorithm, compute shares of the inference result, and send them back to U.

In the threat model adopted in this study, we assume all entities are semi-honest. The following types of attacks are considered: CS may attempt to analyze input data used in the model to collect user private information and infer SVM model parameters; U might deduce model parameters by analyzing the obtained inference results.

Verifiable Privacy-Preserving Inference

Figure 2 presents the system model of our verifiable privacy-preserving inference model. The model comprises three entities: The Model Provider (MP), Cloud Servers (CS), and User (U).

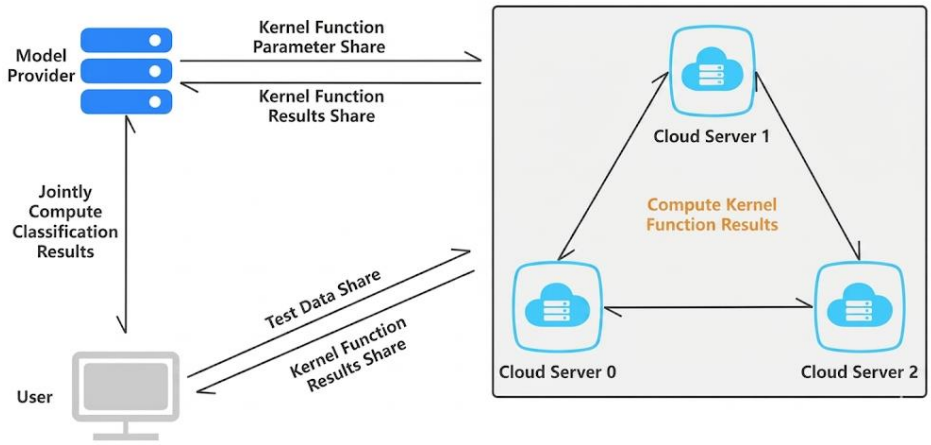

Figure 2. System Model for Verifiable Privacy-Preserving Inference Framework

1. *The Model provider:* In addition to outsourcing parameters to CS, MP require U to jointly verify the correctness of the kernel function and compute the inference result.

2. *User:* In addition to sending the to-be-classified sample to CS, U must also jointly verify the correctness of the kernel function with MP and compute the inference result.

3. *Cloud Servers:* CS only needs to compute the kernel function values and transmit the two-party additive secret-sharing shares of these values to both MP and U.

The detailed workflow of the inference model is structured as follows:

In the preprocessing phase, U sends both the original samples and preprocessed samples along with their three-party secret shares to CS. MP concurrently transmits three-party secret shares of the kernel function parameters and their processed versions to CS, while sending two-party secret shares of other SVM model parameters to U. Upon receiving parameters from MP and samples from U, CS computes kernel function values, converts the replicated secret sharing kernel values into additive secret sharing format, and distributes the transformed shares to MP and U respectively. After obtaining the shares from CS, MP and U jointly verify the correctness of kernel function values to obtain the inference result.

In this established threat model, we assume cloud servers (CS) can be malicious. However, please note that only non collusive malicious cloud servers are allowed to exist. Furthermore, we consider the following attack types: CS may attempt to steal user input samples, extract SVM kernel function parameters, and potentially forge kernel function computation results; U could infer model parameters through analysis of inference results; MP might deduce users' input samples from kernel function values to obtain private information.

BUILDING BLOCKS

This section introduces the security building blocks employed in the protocol, elaborating on their functionalities and operational mechanisms.

Bit2A

In both the comparison function of our outsourcing protocol and the secure compu- tation of kernel functions, all operations must be performed under Boolean sharing within the Z_2 secret sharing domain.

Since subsequent operations (e.g., dot products) demonstrate higher computational efficiency in arithmetic circuits, it becomes essential to convert these Boolean shares into arithmetic shares. The Bit2A conversion is achieved through shared random bits between arithmetic and Boolean circuits [10]. Assuming a random bit r with existing shares $[[r]]$ and $[[r]]^B$ (i.e., shares under Boolean circuits), cloud servers (CS) can securely hide secret bit $[[b]]^B$ using $[[r]]^B$ and publicly reveal $b \oplus r$ without information leakage. Subsequently, the mask $[[r]]$ can be removed from $[[b \oplus r]]$ to recover $[[b]]$.

A2B

A2B [11] performs the inverse function of Bit2A, converting arithmetic shares of $x \in Z_{2^l}$ into their binary shares ($[[x_{l-1}]]^B$, $[[x_0]]^B$). An efficient A2B conversion scheme under Replicated Secret Sharing (RSS) was proposed in [12]. The core idea involves computing $x = x_0 + x_1 + x_2$ within a binary circuit, where each x_i can generate corresponding deterministic binary shares. Executing this addition circuit produces the binary shares of x as output.

• $[[x_i]]^B$, $[[x_i]] \leftarrow PRandBitD()$ The protocol generates a random secret bit shared in F_2 and Z_q.

• $[[x]] \leftarrow PRandInt()$ The effect of this protocol is to generate a random integer.

• $x \leftarrow Output([[x]])$ The effect of this protocol is to decrypt the share and output the plaintext of the share.

• ($[[x_{k-1}]]^B$, ..., $[[x_0]]^B$) $\leftarrow$ $BitAdd((a), ([[b]]^B))$ The effect of this protocol is to calculate the k least significant bits of $[[x]]$ using binary addition.

Algorithm 1：A2B

Input： $[[x]]$

Output： $[[x]]^B$

1：for $i = 0 \to l - 1$ do:

2: $[[r_i]]^B, [[r_i]] \leftarrow PRandBitD();$

3：$[[r]]' \leftarrow \sum_{i=0}^{l-1} 2^i * [[r_i]];$

4：$[[r]]'' \leftarrow PRandInt();$

5：$[[r]] = 2^m \cdot [[r]]'' + [[r]]';$

6：$c = Output([[x]] - [[r]]);$

7：$([[x_{l-1}]]^B, \cdots, [[x_0]]^B) \leftarrow BitAdd((c_{l-1}, \cdots, c_0), ([[r_{l-1}]]^B, \cdots, [[r_0]]^B));$

8：return $\left[[x_{l-1}]\right]^B, \cdots, \left[[x_0]\right]^B$;

Secure Comparison

We implement comparison operations by computing the difference between two operands, where the comparison result is determined by the sign of the difference.

Since the sign of ring elements can be determined by their most significant bit (MSB)[13], MSB can thus represent the comparison outcome. As outlined in Algorithm 2, given two operands *x* and *y*, the secret is shared. The cloud server first computes the difference locally [[*a*]] = [[*x*]] − [[*y*]]. Then, it applies A2B to determine the highest significant bit $[[b]]^B$ of [[*a*]] and subsequently uses Bit2A to convert $[[b]]^B$ to [[*b*]]. This protocol is applicable to both two-party additive secret sharing shares and three-party replicated secret sharing shares.

Algorithm 2：Comparison

Input： $\left[[\mathrm{x}]\right], [[\mathrm{y}]]$

Output： $[[\mathrm{b}]]$

1：Locally compute $\left[[a]\right] = \left[[x]\right] - [[y]]$;

2：Compute $[[b]]^B = [\left[a[l-1]\right]]^B$;

3：$[[b]] \leftarrow Bit2A([[b]]^B)$;

4：return $[[b]]$;

PROTOCOL

In this section we present each of the two different cloud-based privacy-preserving outsourcing protocols we propose.

Efficient Privacy-Preserving Inference

We first introduce our efficient privacy-preserving inference model (*OPVSVM*). First, we elucidate the secure computation mechanism through which CS computes kernel function values. After completing the kernel value computation, decision function val- ues for all classes of the sample are calculated based on these computational results. We then demonstrate how the inference result for this specific sample is

generated by analyzing these decision function values. Finally, the comprehensive workflow of the entire inference model is introduced.

Kernel Function

Algorithm 3 elaborates how to compute the kernel function. The function inputs are shares of support vectors *x* and shares of the sample *t*, denoted as $\langle x \rangle$ and $\langle t \rangle$Since our model employs a Gaussian kernel function, the first step involves computing the difference $\langle h \rangle$ between $\langle x \rangle$ and $\langle t \rangle$. Next, the power part of the Gaussian kernel $\langle H \rangle$ is then computed by $\langle h \rangle$ and the parameter λ specified in advance. Subsequently, an A2B conversion is applied to transform $\langle H \rangle$ into its Boolean sharing representation. For each bit of the Boolean shares, we perform a Bit2A conversion to obtain arithmetic shares $\langle H \rangle$. Finally, bitwise exponential operations are performed on these arithmetic shares to obtain the Gaussian kernel function shares [13].

Algorithm 3：Kernel

Input： $< \boldsymbol{x} >, < \boldsymbol{t} >$

Output： $< Kernel >$

1：$< h >=< x > - < t >$;

2：$< H >=< h > \times < h > \times (-\lambda)$;

3：$< H_0 >^B, \cdots, < H_{l-1} >^B \leftarrow A2B(< H >)$;

4：for $i = 0 \rightarrow l - 1$ do

5：$< H_i > \leftarrow Bit2A(< H_i >^B)$;

6：$< Kernel > \leftarrow \prod_{i=0}^{l-1}(< H_i > \times 2^i + 1 - < H_i >)$;

7：return $< Kernel >$;

Secure Data Reading

To conceal inference results during max-class identification, we devise a secure data retrieval protocol that enables private access to target data through secret-shared position indices. Given a secret-shared array [[L]] and secret-shared target index [[b[m−1]]], the protocol proceeds as follows: First, apply A2B conversion to decompose [[b[m−1]]] into its bitwise representation, ensuring exactly one bit position contains 1. Subsequently, transform these bits into a bit vector via Demux function, which significantly enhances computational efficiency in secure computation frameworks. Finally, perform element-wise se-

cure multiplication between the bit vector and array [[L]], accumulating the results through iterative secure multiplications to obtain the desired value. [[L]] ← Demux([[b[m−1]l−1]]B, . . . , [[b[m−1]0]]B) The effect of this protocol is to convert data in bit-decomposition format into bit-vector form [14].

Algorithm 4：SecRead

Input： $[[L]], [[\mathrm{b}[\mathrm{m}-1]]]$

Output： $[[L_{max}]]$

1：$([[b[m-1]_0]]^B, \cdots, [[b[m-1]_{l-1}]]^B) \leftarrow A2B([[b[m-1]]])$;

2：$[bitv] = Demux([b[m-1]_{l-1}]^B, \cdots, [b[m-1]_0]^B)$;

3：$[L_{max}] = 0$;

4：for $i = 0 \rightarrow l-1$ do

5：$[L_{max}] = [L_{max}] + [bitv_i] * [L_i]$;

6：return $[[L_{max}]]$;

Maximum-class Identification

Algorithm 5 delineates the procedure for obtaining the final inference result. The input to this algorithm is the shares of decision function values ⟨*Di*⟩ and class labels ⟨*Li*⟩ for the unclassified sample across. Generate two blinded arrays ⟨*a*⟩ and ⟨*b*⟩ of length *m*. Initially, pairwise comparisons are performed between the decision function values of different classes, specifically comparing ⟨*Di*⟩ and ⟨*Di*+1⟩ iteratively. By invoking the secure comparison function, only the share form of the result ⟨*a*[*i*]⟩ is returned without revealing the actual com- parison outcome. The computation of ⟨*a*[*i*]⟩ · ⟨*Di*⟩ +(1__ ⟨*a*[*i*]⟩ .) · ⟨*Di*+1⟩ enables ⟨*Di*+1⟩ to be dynamically updated with the maximum value between ⟨*Di*⟩ and ⟨*Di*+1⟩. Con- currently, ⟨*b*[*i* + 1]⟩ stores the index of the larger value. After iterating through all classes, ⟨*b*[*m*]⟩ contains the index corresponding to the class with the maximum deci- sion function value. The label of this maximum class is ultimately retrieved based on the stored index.

Algorithm 5：ASSSortMax

Input： $< D >, < \mathrm{L} >$

Output： $< \boldsymbol{L_{max}} >$

1：Sets vectors $< a >$ and $< b >$ of length m；

2: for $i = 0 \to m - 2$ do
3: $< \text{a[i]} > \leftarrow Comparison(< D_i >, < D_{i+1} >)$;
4: $< D_{i+1} > = < D_i > \times < \text{a[i]} > + < D_{i+1} > \times < \text{a[i} + 1] >$;
5: $< b[i+1] > = i \times < a[i] > + (i+1) \times (1 - < \text{a[i]} >)$;
6: $< \boldsymbol{L_{max}} > \leftarrow SecRead(< \text{L} >, < b[m-1] >)$;
7: return $< \boldsymbol{L_{max}} >$;

Complete Algorithm

This section delineates the complete workflow of our efficient privacy-preserving infer- ence model. Prior to initiating the algorithm, U is assumed to possess the unclassified sample t, while MP maintains the complete multi-class SVM classifier parameters such as $C_{i,j}$, b_i, $x_{i,j}$, L_i. Initially, U and MP convert all data into three-party repli- cated secret shares and transmit them to the CS. After CS receives all parameters, it iterates through all categories and their corresponding support vectors, invokes the Kernel function to compute the corresponding kernel values. Utilizing these kernel val- ues, CS then calculates the decision function value D_i for each class regarding sample t. Ultimately, the system securely identifies the label corresponding to the maximum decision function value, which constitutes the final inference result.

Verifiable Privacy-Preserving Inference

We propose validated privacy preserving inference model in the presence of verifica- tion mechanisms for possible malicious cloud servers. The kernel function part of the protocol are the same as in Algorithm 3. The maximum-class identification compo- nent of the protocol is demonstrated in the Algorithm 7. The main difference from the previous maximum-class identification protocol is that all operations are performed using two-party secret sharing instead of three-party secret sharing.

Algorithm 6: OPVSVM

Input: U inputs $\boldsymbol{t}$, MP inputs $\boldsymbol{C_{i,j}}, \boldsymbol{b_i}, \boldsymbol{x_{i,j}}, \boldsymbol{L_i}$
Output: $< \boldsymbol{L} >$
1: U generates sharing $< t >$;
2: $< t > \to CS$;
3: MP generates sharing $< \boldsymbol{x_{i,j}} >, < C_{i,j} >, < b_i >, < \boldsymbol{L_i} >$;
4: $< \boldsymbol{x_{i,j}} >, < C_{i,j} >, < b_i >, < \boldsymbol{L_i} > \to$ CS;
5: CS: for $i = 0 \to m - 1$ do
6: for $j = 0 \to s_i$ do

7： $< k_{i,j} >= Kernel(< \boldsymbol{x}_{i,j} >, < t >)$；

8：for $i = 0 \rightarrow m - 1$ do

9： $< D_i >= \sum_{j=0}^{k} < C_{i,j} > \times < k_{i,j} > + < b_i >$；

10：$< L >= ASSSortMax(< D_i >, < L_i >)$；

11：return $< L >$；

Algorithm 7：RSSSortMax

Input： $\left[[\boldsymbol{D}]\right], [[L]]$

Output： $[[\boldsymbol{L_{max}}]]$

1：Sets vectors $[[a]]$ and $[[b]]$ of length m；

2：for $i = 0 \rightarrow m - 2$ do

3： $[[\mathrm{a[i]}]] \leftarrow Comparison([[D_i]], [[D_{i+1}]])$；

4: $\left[[D_{i+1}]\right] = \left[[D_i]\right] \times \left[\left[\mathrm{a[i]}\right]\right] + \left[[D_{i+1}]\right] \times [\left[\mathrm{a[i+1]}\right]]$；

5： $\left[\left[b[i+1]\right]\right] = i \times [\left[a[i]\right]] + (i+1) \times (1 - [\left[\mathrm{a[i]}\right]])$；

6：$[[\boldsymbol{L_{max}}]] \leftarrow SecRead([[\mathrm{L}]], [[b[m-1]]])$；

7：return $[[\boldsymbol{L_{max}}]]$;

Decision Function

Algorithm 8 implements decision function computation for the i-th class while con- currently conduct-ing malicious cloud server verification. The inputs are blinded kernel function values $[[K']]$ and non-blinded kernel function values $[[k]]$. First, validate whether $([[k_{i,j}]] \times [[\_\_e^{r}]] + [[k'_{i,j}]])! = 0$ holds. If the condition is violated, malicious cloud servers are detected. Upon successful validation, the decision function value for the i-th class is computed and output.

Algorithm 8： Decision

Input： $\left[[k_{i,j}]\right], \left[[k'_{i,j}]\right]$

Output： $[[D_i]]$

1：for j= $0 \rightarrow s_i$ do

2： if $\left(\left[[k_{i,j}]\right] \times \left[[-e^{r^2}]\right] + \left[[k'_{i,j}]\right]\right)! = 0$ then：

3： return $[[D_i]] = 0$；

4：$[[D_i]] = \sum_{j=0}^{m}\left[[C_{i,j}]\right] \times \left[[k_{i,j}]\right] + [[b_i]]$；

5：return $\left[[D_i]\right]$;

Complete Algorithm

We maintain the assumption that U possesses the sample t and MP holds the SVM model parameters. Initially, U generates a blinding factor r to produce the blinded sample t'. Shares $\langle t \rangle$, $\langle t' \rangle$, and $[[\_e^r]]$ are generated and distributed to CS and MP according to the procedure outlined in Algorithm 10. Upon receiving r, MP blinds the

support vectors and transmits shares of the original $x_{i,j}$ and blinded $x_{i,j}$ to CS, while sending shares of other parameters to U.

Subsequently, U and MP interactively execute decision function evaluation and perform adversarial verification on cloud servers, finally obtaining the inference result through secure computation protocols that maintain privacy-preserving properties during collaborative verification. The detailed workflow is specified in Algorithm 9.

SECURITY ANALYSIS

In our model, many building blocks are used to ensure the security of the protocol. Therefore, to establish the security of the protocol, we first need to prove the security of our building blocks. Since the security of the components we use, such as A2B and Bit2A, has already been proven in previous work, we do not repeat the security proofs for these existing components.

The maximum-class identification module operates on secret-shared representations of decision function values. The protocol executes secure pairwise comparisons between adjacent decision values ([[D_i]] and [[D_i+1]]) in sequence. After each comparison, [[D_i+1]] is updated to retain the greater value while recording the corresponding class index in array b. Upon completing all comparisons, the final decision value represents the maximum value, with the last entry in b containing the index of the identified maximum

class. The identification result is then derived from this index. Throughout this process, all decision values, intermediate comparison results, the maximum value, and class indices remain protected under the simulation-based security framework, thereby achieving provable security.

Algorithm 8：WPVSVM

Input： U inputs $\boldsymbol{t}$， MP inputs $\boldsymbol{C_{i,j}}, \boldsymbol{b_i}, \boldsymbol{x_{i,j}}, \boldsymbol{L_i}$

Output： $[\![L]\!]$

1：U randomly chooses r；

2：$t' = t \times r$；

3：Generates sharing $< t >, < t' >, [\![-e^{r^2}]\!]$；

4：$< t >, < t' > \rightarrow CS; r, [\![-e^{r^2}]\!] \rightarrow MP$；

5：MP: $x'_{i,j} = x_{i,j} \times r$；

6：Generates sharing $< x_{i,j} >, < x'_{i,j} >, [\![C_{i,j}]\!], [\![b_i]\!], [\![L_i]\!]$；

7：$< x_{i,j} >, < x'_{i,j} > \rightarrow CS$， $[\![C_{i,j}]\!], [\![b_i]\!], [\![L_i]\!] \rightarrow U$；

8：CS: for $i = 0 \rightarrow m - 1$ do

9： for $j = 0 \rightarrow s_i$ do

10： $< k_{i,j} >= Kernel(< x_{i,j} >, < t >), < k_{i,j}{}' >= Kernel(< x_{i,j}{}' >, < t' >)$；

11： $< k_{i,j} >, < k_{i,j}{}' > \rightarrow U$；

12：$< k_{i,j} >, < k_{i,j}{}' > \rightarrow [\![k_{i,j}]\!], [\![k_{i,j}{}']\!]$；

13：$[\![k_{i,j}]\!], [\![k_{i,j}{}']\!] \rightarrow U, MP$；

14：for $i = 0 \rightarrow m - 1\ do$

15： $[\![D_i]\!] = Decision([\![k_{i,j}]\!], [\![k_{i,j}{}']\!])$；

16： if $[\![D_i]\!] == 0$ then

17： return false;

18：$[\![L_{max}]\!] = RSSSortMax([\![D]\!], [\![L]\!])$;

19：return $[\![L_{max}]\!]$;

In our designed verification mechanism, U and MP first send two different parameters (one blinded and one non-blinded) to CS. CS cannot distinguish which parameter is blinded from the received data, thereby ensuring that the blinded parameters are not acquired by CS. When computing the kernel functions, CS will separately generate the non-blinded kernel function $k_{i,j}$,and the blinded kernel function $k'_{i,j}$, as shown below:

$$k_{i,j} = e^{-\lambda \cdot \|x_{i,j}-t\|_2^2}$$

$$k'_{i,j} = e^{-\lambda \cdot \|x_{i,j}*r-t*r\|_2^2} = k_{i,j} * e^{r^2} \quad (4)$$

Therefore, we only need to verify whether $k'_{i,j} != k_{i,j} * e^{r}$ to determine the correct- ness of the obtained results. In this process, all parameters and intermediate values are computed in the form of secret shares, effectively ensuring the privacy of the data. In our efficient privacy-preserving inference model, all computational processes are executed on cloud servers. From the perspective of each cloud server, input parameters and intermediate values during computation remain in secret-shared form throughout,

where the security of the protocol is guaranteed by the security properties of Replicated Secret Sharing (RSS).

For the verifiable privacy-preserving inference model, the protocol first securely computes kernel function values in the cloud. This step's security is similarly ensured by the RSS scheme. After converting the kernel function values to additive secret- sharing format, the subsequent processes for securely obtaining inference results by participants U and MP are protected by the security guarantees of two-party additive secret sharing.

EVALUATION

In this section, we evaluate the performance of our proposed system in terms of communication overhead and time overhead by comparing it with other schemes.

Initialisation

Our experiments were conducted on a Ubuntu-based desktop computer equipped with an Intel core i7-11700K CPU 3.60GHz*x*4 and 64-GB RAM. We evaluated the performance of the program based on the MNIST dataset. We implemented a proof-of-concept prototype. The evaluation on the MNIST dataset validates the fundamental correctness and establishes a baseline for the cryptographic overhead of our secure SVM inference scheme. While CPS domains encompass highly diverse and complex data structures, we uti-

lize the MNIST dataset in this section as a standard, widely-recognized cryptographic benchmark. This approach allows us to isolate and evaluate the computational and communication overhead introduced by our secure multiparty computation protocols, rather than the predictive capacity of the SVM model itself.

The MNIST dataset is a dataset extensively used in image inference tasks, particularly holding significant importance in the field of handwritten digit recognition. Comprising 60,000 training images and 10,000 test images, each sample is a 28×28 pixel grayscale image representing handwritten digits from 0 to 9. The dataset originates from U.S. postal code forms across various regions, having undergone normalization and noise reduction processes to ensure image clarity and standardization.

Our implementation leverages the MP-SPDZ framework, an open-source plat- form supporting multi-party computation (MPC) protocols. This framework provides a comprehensive benchmarking suite for secure programs through its integration of generic MPC primitives.

We compare the two proposed schemes, *WPVSVM* and *OPVSVM*, with recently proposed privacy-preserving machine learning prediction schemes, respectively. Where *WPVSVM* is compared with *PVSSVM* [9] and *OPVSVM* is compared with *OSSVM*[15].

Computational Overhead

We will investigate the impact of these two aspects on computational overhead by changing both the feature dimensions of support vectors and the training set sizes. Employing the control variable method, we first vary the feature dimensions from 50 to 250 (while keeping the training set size fixed at 200) to compare the computational overhead between our scheme and the comparison scheme under increasing feature dimensions. Subsequently, we adjust the training set sizes from 200 to 1000 (while maintaining the feature dimensions constant at 50) to perform comparative analysis with the comparison scheme.

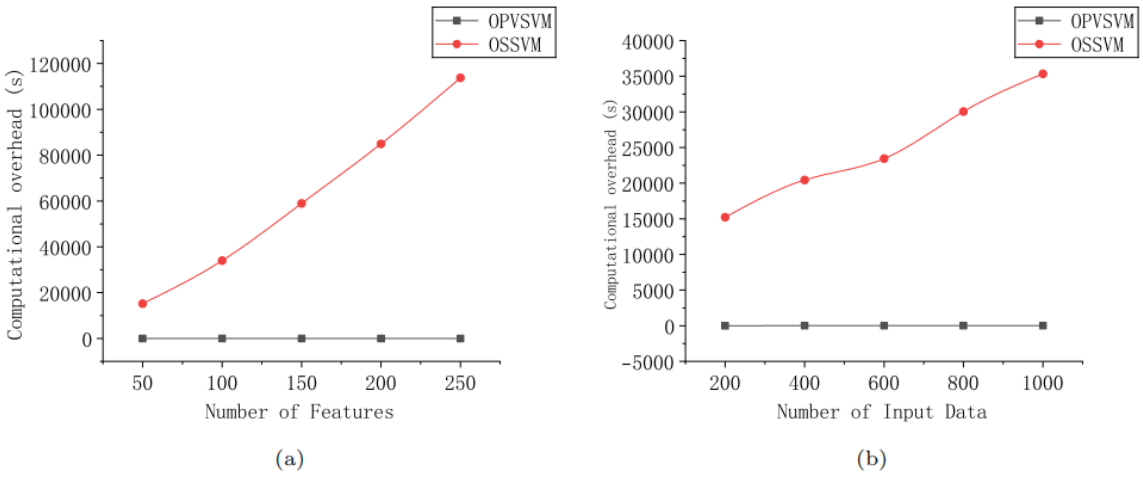

Figure 3. Computational overhead of DFGen

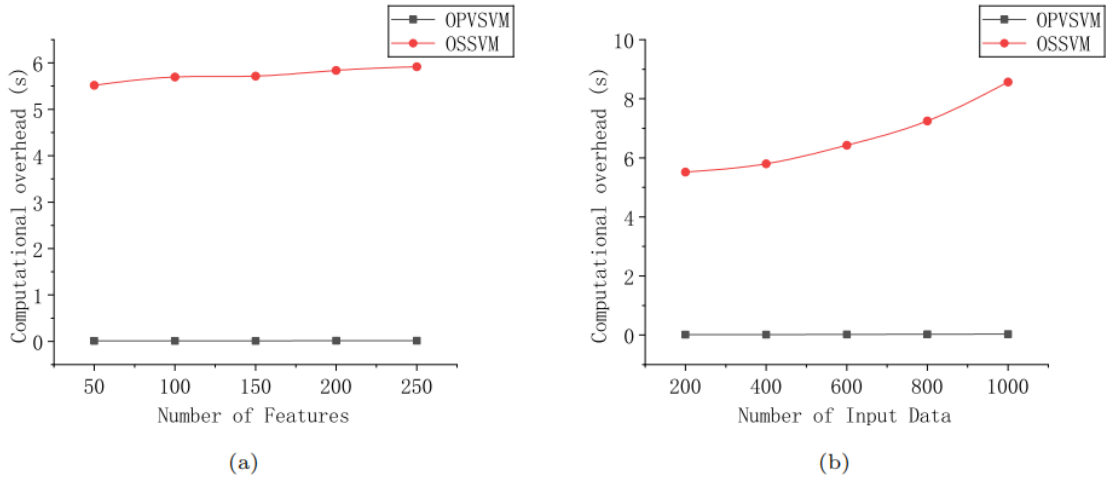

Figure 4. Computational overhead of IRGen

For our *OPVSVM*, the evaluation will be conducted in two phases: the Decision Func- tion Generation phase (DFGen) and the Inference Result Generation phase (IRGen). As shown in Figure 3 and 4, the computational overhead of both schemes increases with feature dimension growth, yet our scheme demonstrates signif- icantly lower overhead compared to *OSSVM*. For instance, with a training set size of 200 samples and feature dimension of 50, *OSSVM* requires 5.515 seconds to identify the maximum-class category, where as *OPVSVM* completes this process in merely 0.016 seconds. Figure 3 and 4 also illustrate the comparative

computational overhead between *OPVSVM* and *OSSVM* under increasing training set sizes, where *OPVSVM* consistently exhibits superior efficiency.

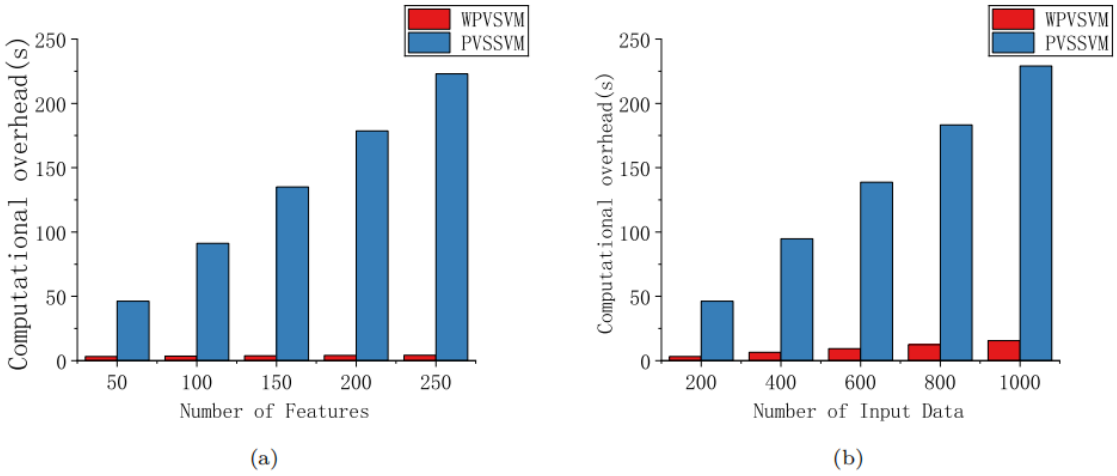

Figure 5. Computational overhead of DFGen

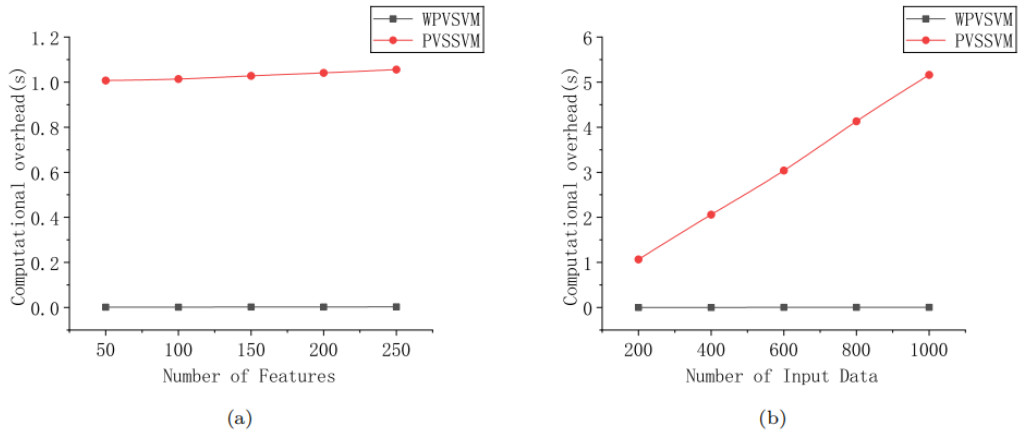

Figure 6. Computational overhead of IRGen

WPVSVM

We divide *WPVSVM* into three phases: the Decision Function Generation phase (DFGen) and Inference Result Generation phase (IRGen) shared with *OPVSVM*, along with a newly introduced Result Verification phase (VerGen). Figure 5 demon- strates the computational advantages of our DFGen phase. While *PVSSVM* employs Shamir's secret sharing, our approach uses three-party replicated secret sharing under

the three-cloud model, which exhibits significant computational advantages during dot product operations. Figure 6 illustrates the IRGen phase, while Figure 7 compares the

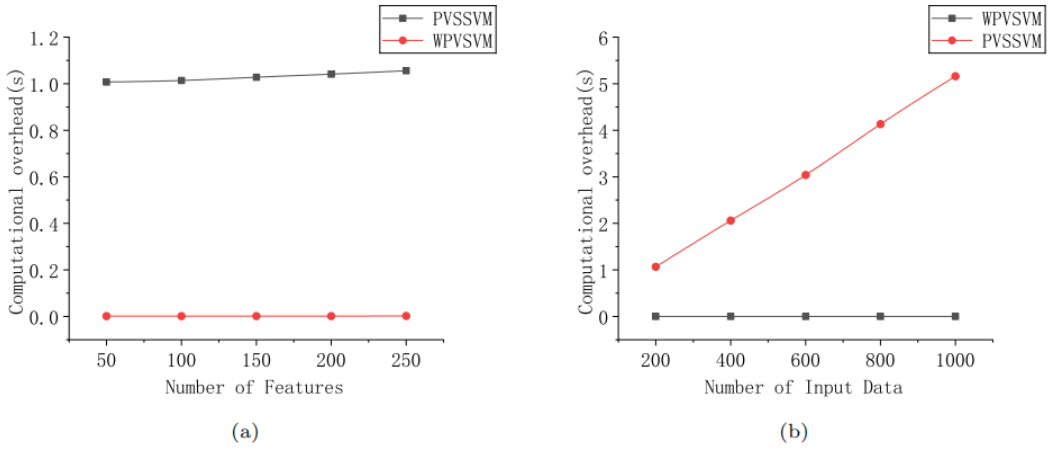

Figure 7. Computational overhead of VerGen

VerGen phase. With fixed input feature dimensions of 50 and a training set size of 400, *PVSSVM* consumes 94.78 seconds in DFGen and 2.05958 seconds in VerGen. In contrast, *WPVSVM* requires only 41.988 seconds for DFGen and 0.00011 seconds for VerGen.

Communication Overhead

The communication cost of our proposed scheme primarily stems from distributing shared data, which becomes particularly pronounced under large-scale datasets and high-dimensional features. Figure 8,9 and 10 demonstrate different phases communication overhead of our two protocols under varying input data quantities and feature dimensions. It is evident that our communication overhead grows with both increasing feature dimensions and input data quantities, but exhibits significantly faster growth rates with the latter. This indicates that data quantity exerts a more dominant influ- ence on communication costs in our scheme. Furthermore, the communication cost incurred during the lookup phase solely depends on the input data quantity and remains independent of feature dimension size.

RELATED WORK

In privacy-preserving machine learning, researchers have developed various schemes to protect sensitive data during Support Vector Machine (SVM) inference processes, with applications in medical diagnosis [16-19], image recognition [20-22], financial analysis [23-[25], and text recognition [26-29]. For instance, in the medical diagnosis domain [30-32], these schemes [33-36] protect users' health information. Early work focused on training SVM classifiers in distributed environments where training samples were partitioned across mutually distrustful par- ties. These protocols enabled collaborative classifier construction without disclosing local data [37-38]. During inference, participants holding partial classifier parame- ters jointly computed decision functions to output results. Yu et al.[37] and Hu et al.[38] investigated privacy-preserving SVM inference for horizontally and vertically partitioned data, respectively.

Cloud computing has emerged as the dominant trend for future data service out- sourcing due to its elimination of high operational costs and expensive hardware procurement. However, the incomplete trustworthiness of third-party cloud service providers makes preventing privacy data leakage a critical challenge. Zhu et al. [39] proposed a medical data privacy-preserving scheme, enabling cloud servers to per- form inference computations on blinded datas while restricting participants to partial blinded data access. Xie et al. [15] designed a two-cloud non-collaborative privacy- preserving online diagnostic system, randomly partition SVM parameters, user inputs, and diagnostic results across two servers, ensuring neither server obtains complete information. Zhang et al. [40] developed a multi-class linear SVM online diagnostic system, but its maximum decision function search process risks exposing access pat- terns, resulting in insufficient protection of inference results. Compared to the above schemes, *OPVSVM* achieves complete protection of data privacy while optimising model performance and improving response efficiency.

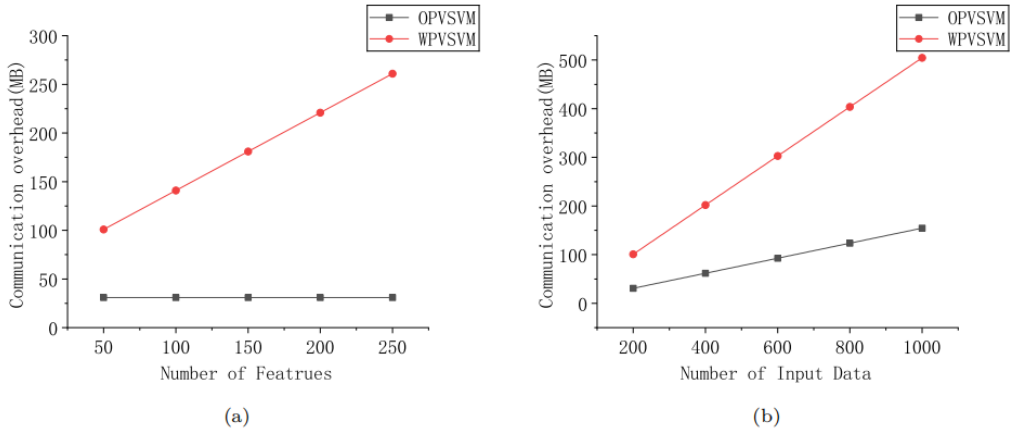

Figure 8. Communication overhead of DFGen

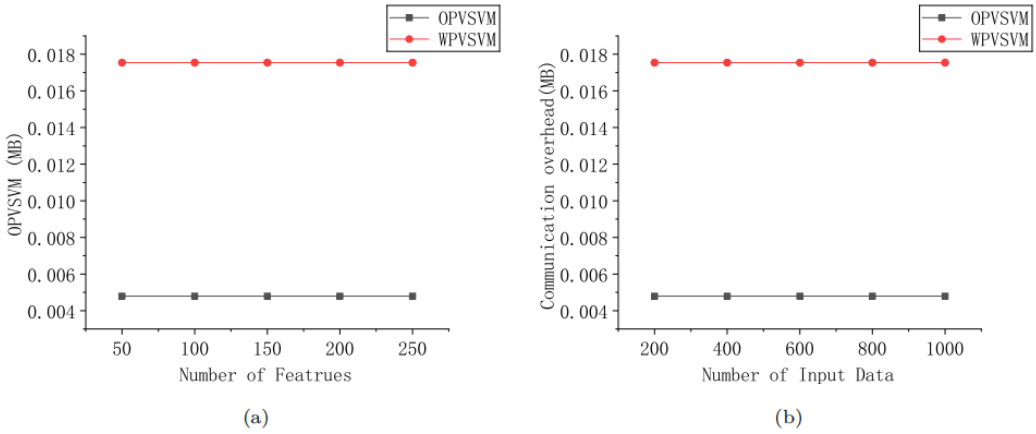

Figure 9: Communication overhead of IRGen

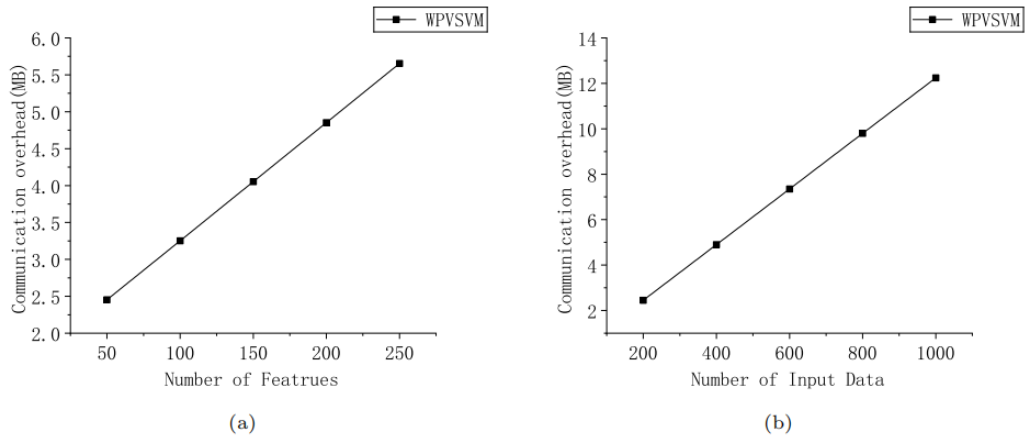

Figure 10. Communication overhead of VerGen

The above schemes are primarily designed under the assumption of semi-honest cloud servers. However, real-world scenarios often involve malicious cloud servers. To address potential output tampering by malicious servers, verification mechanisms can be incorporated to ensure the correctness and integrity of results. Liang et al. [4] implemented SVM inference using OPE, though this approach risks exposing sensitive data ordering patterns. Lei et al. [9] used the method of adding labels to the data in the inference process to propose a corresponding mechanism for validating the inference results based on improved efficiency. Unlike prior schemes, our scheme can both ensure the computational efficiency and complete the validation of the inference results on the basis of data security. Unlike previous schemes, the verification mechanism of *WPVSVM* balances privacy and efficiency both without the risk of data leakage [4] and without the use of polynomial interpolation [9], which is a relatively time- consuming verification method.

CONCLUSION

In this paper, we first present *OPVSVM*, an efficient privacy-preserving inference model that achieves exceptional operational efficiency while maintaining rigorous security guarantees. Additionally, we propose *WPVSVM*, a verifiable privacy-preserving inference model incorporating verification mechanisms to effectively address malicious cloud server scenarios and validate result authenticity. From a practical implementation standpoint, we adopt replicated secret sharing technology to instantiate our framework. Furthermore, our secure maximum-class identification function executes inference procedures with O(n) time complexity without introducing information leakage. Experimental evaluations demonstrate that both models exhibit superior performance compared to schemes in the same type. While our baseline evaluation on MNIST confirms the theoretical efficiency of the proposed framework, we acknowledge that it does not fully represent the high-dimensional, time-series, or heterogeneous data typical in actual AI-enabled CPS (such as smart healthcare or critical infrastructure). Therefore, a critical direction for our future work involves deploying and benchmarking the framework using domain-specific datasets (e.g., real-time medical sensor data or industrial control logs) to rigorously evaluate its practical real-time performance and scalability under strict CPS constraints.

Author Contributions

Conceptualization – Shimao Yu; methodology – Shimao Yu; formal analysis – Shimao Yu; investigation – Shimao Yu; resources – Shimao Yu; writing-original draft preparation – Shimao Yu; writing-review and editing – Shimao Yu; visualization – Shimao Yu; supervision – Shimao Yu. All authors have read and agreed to the published version of the manuscript.

Conflicts of Interest

The authors declare no conflict of interest.

Funding

This research received no external funding.

Acknowledgements

Not applicable.

REFERENCES

[1] Zhang Z, Wang N, Wu H, Tang C, Li R. MR-DRO: A fast and efficient task offloading algorithm in heterogeneous edge/cloud computing environments. IEEE Internet of Things Journal. 2023; 10(4):3165–3178. https://doi.org/10.1109/JIOT.2021.3126101

[2] Zhang P, Chui Y, Liu H, Yang Z, Wu D, Wang R. Efficient and privacy-preserving search over edge–cloud collaborative entity in IoT. IEEE Internet of Things Journal. 2023; 10(4):3192–3205. https://doi.org/10.1109/JIOT.2021.3132910

[3] Yunhong H, Liang F, Guoping H. Privacy-preserving svm classification on vertically partitioned data without secure multi-party computation. 2009 Fifth International Conference on Natural Computation. 2009; 1:543–546. https://doi.org/10.1109/ICNC.2009.120

[4] Liang J, Qin Z, Xiao S, Zhang J, Yin H, Li K. Privacy-preserving range query over multi-source electronic health records in public clouds. Journal of Parallel and Distributed Computing. 2020; 135:127–139. https://doi.org/10.1016/j.jpdc.2019.08.011

[5] Li X, Zhu Y, Wang J, Liu Z, Liu Y, Zhang M. On the soundness and security of privacy-preserving svm for outsourcing data classification. IEEE Transactions on Dependable and Secure Computing. 2018; 15(5):906–912. https://doi.org/10.1109/TDSC.2017.2682244

[6] Zhang M, Song W, Zhang J. A secure clinical diagnosis with privacy-preserving multiclass support vector machine in clouds. IEEE Systems Journal. 2022; 16(1):67–78. https://doi.org/10.1109/JSYST.2020.3027758

[7] Liang J, Qin Z, Xue L, Lin X, Shen X. Verifiable and secure svm classification for cloud-based health monitoring services. IEEE Internet of Things Journal. 2021; 8(23):17029–17042. https://doi.org/10.1109/JIOT.2021.3075540

[8] Niu C, Wu F, Tang S, Ma S, Chen G. Toward verifiable and privacy preserving machine learning prediction. IEEE Transactions on Dependable and Secure Computing. 2022; 19(3):1703–1721. https://doi.org/10.1109/TDSC.2020.3035591

[9] Lei D, Liang J, Zhang C, Liu X, He D, Zhu L, Guo S. Publicly verifiable and secure svm classification for cloud-based health monitoring services. IEEE Internet of Things Journal. 2024; 11(6):9829–9842. https://doi.org/10.1109/JIOT.2023.3326358

[10] Rotaru D, Wood T. MArBled Circuits: Mixing Arithmetic and Boolean Circuits with Active Security. Cryptology ePrint Archive. 2019; 2019:207. https://eprint.iacr.org/2019/207

[11] Escudero D, Ghosh S, Keller M, Rachuri R, Scholl P. Improved primitives for MPC over mixed arithmetic-binary circuits. Lecture Notes in Computer Science. 2020; 12171:823-852. https://doi.org/10.1007/978-3-030-56880-1_29

[12] Mohassel P, Rindal P. Aby3: A mixed protocol framework for machine learning. Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. 2018; :35–52. https://doi.org/10.1145/3243734.3243760

[13] Keller M, Sun K. Secure Quantized Training for Deep Learning. arXiv. 2022; https://arxiv.org/abs/2107.00501

[14] Keller M, Scholl P. Efficient, oblivious data structures for mpc. Advances in Cryptology – ASIACRYPT 2014. 2014; :506–525. https://doi.org/10.1007/978-3-662-45608-8_27

[15] Xie B, Xiang T, Liao X, Wu J. Achieving privacy-preserving online diagnosis with outsourced svm in internet of medical things environment. IEEE Transactions on Dependable and Secure Computing. 2022; 19(6):4113–4126. https://doi.org/10.1109/TDSC.2021.3119897

[16] Arora G, Dubey AK, Jaffery ZA, Rocha A. Bag of feature and support vector machine based early diagnosis of skin cancer. Neural Computing and Applications. 2022; 34:12669–12676. https://doi.org/10.1007/s00521-022-07132-5

[17] Hao J, Luo S, Pan L. Rule extraction from biased random forest and fuzzy support vector machine for early diagnosis of diabetes. Scientific Reports. 2022; 12(1):9858. https://doi.org/10.1038/s41598-022-14143-8

[18] Wang J, He F, Sun S. Construction of a new smooth support vector machine model and its application in heart disease diagnosis. Plos one. 2023; 18(2):0280804. https://doi.org/10.1371/journal.pone.0280804

[19] Habib M, Wang Z, Qiu S, Zhao H, Murthy AS. Machine learning based healthcare system for investigating the association between depression and quality of life. IEEE Journal of Biomedical and Health Informatics. 2022; 26(5):2008–2019. https://doi.org/10.1109/JBHI.2021.3121544

[20] Li Y, Li J, Pan JS. Hyperspectral image recognition using svm combined deep learning. J. Internet Technol. 2019; 20(3):851–859. https://doi.org/10.3966/160792642019052003017

[21] Jiang F, Lu Y, Chen Y, Cai D, Li G. Image recognition of four rice leaf diseases based on deep learning and support vector machine. Computers and Electronics in Agriculture. 2020; 179:105824. https://doi.org/10.1016/j.compag.2020.105824

[22] Chandra MA, Bedi S. Survey on svm and their application in image classification. International Journal of Information Technology. 2021; 13(5):1–11. https://doi.org/10.1007/s41870-017-0080-1

[23] Islam SM, Rahman A, Prasad N, Boric-Lubecke O, Lubecke VM. Identity authentication system using a support vector machine (svm) on radar respiration measurements. 2019 93rd ARFTG Microwave Measurement Conference (ARFTG). 2019; :1–5. https://doi.org/10.1109/ARFTG.2019.8739226

[24] Lin WH, Wang P, Tsai CF. Face recognition using support vector model classifier for user authentication. Electronic Commerce Research and Applications. 2016; 18:71–82. https://doi.org/10.1016/j.elerap.2016.06.005

[25] Jonsson K, Kittler J, Li Y, Matas J. Support vector machines for face authentication. Image and Vision Computing. 2002; 20(5-6):369–375. https://doi.org/10.1016/S0262-8856(01)00076-0

[26] Tong S, Koller D. Support vector machine active learning with applications to text classification. Journal of machine learning research. 2001; 2(Nov):45–66. https://doi.org/10.1162/153244302760200684

[27] Hamdan YB, Sathesh A. Construction of statistical svm based recognition model for handwritten character recognition. Journal of Information Technology. 2021; 3(02):92–107. https://doi.org/10.36548/jit.2021.2.002

[28] Francis LM, Sreenath N. Robust scene text recognition: using manifold regularized twin-support vector machine. Journal of King Saud University-Computer and Information Sciences. 2022; 34(3):589–604. https://doi.org/10.1016/j.jksuci.2019.11.011

[29] Li H. Text recognition and classification of english teaching content based on svm. Journal of Intelligent & Fuzzy Systems. 2020; 39(2):1757–1767. https://doi.org/10.3233/JIFS-191599

[30] Wang J, Wu L, Wang H, Choo KKR, He D. An efficient and privacy-preserving outsourced support vector machine training for internet of medical things. IEEE Internet of Things Journal. 2020; 8(1):458–473. https://doi.org/10.1109/JIOT.2020.3005898

[31] Liu L, Chen R, Liu X, Su J, Qiao L. Towards practical privacy-preserving decision tree training and evaluation in the cloud. IEEE Transactions on Information Forensics and Security. 2020; 15:2914–2929. https://doi.org/10.1109/TIFS.2020.2979201

[32] Samanthula BK, Elmehdwi Y, Jiang W. K-nearest neighbor classification over semantically secure encrypted relational data. IEEE transactions on Knowledge and data engineering. 2014; 27(5):1261–1273. https://doi.org/10.1109/TKDE.2014.2364027

[33] Liang J, Qin Z, Xiao S, Ou L, Lin X. Efficient and secure decision tree classification for cloud-assisted online diagnosis services. IEEE Transactions on Dependable and Secure Computing. 2019; 18(4):1632–1644. https://doi.org/10.1109/TDSC.2019.2941484

[34] Liang J, Qin Z, Ni J, Lin X, Shen X. Practical and secure svm classification for cloud-based remote clinical decision services. IEEE Transactions on Computers. 2020; 70(10):1612–1625. https://doi.org/10.1109/TC.2020.3013892

[35] Rahulamathavan Y, Veluru S, Han J, Li F, Rajarajan M, Lu R. User collusion avoidance scheme for privacy-preserving decentralized key-policy attribute-based encryption. IEEE Transactions on Computers. 2015; 65(9):2939–2946. https://doi.org/10.1109/TC.2015.2479599

[36] Chen X, Li J, Weng J, Ma J, Lou W. Verifiable computation over large database with incremental updates. IEEE transactions on Computers. 2015; 65(10):3184–3195. https://doi.org/10.1109/TC.2015.2510652

[37] Yu H, Jiang X, Vaidya J. Privacy-preserving svm using nonlinear kernels on horizontally partitioned data. Proceedings of the 2006 ACM Symposium on Applied Computing. 2006; :603–610. https://doi.org/10.1145/1141277.1141415

[38] Yunhong H, Liang F, Guoping H. Privacy-preserving svm classification on vertically partitioned data without secure multi-party computation. 2009 Fifth International Conference on Natural Computation. 2009; 1:543–546. https://doi.org/10.1109/ICNC.2009.120

[39] Zhu H, Liu X, Lu R, Li H. Efficient and privacy-preserving online medical prediagnosis framework using nonlinear svm. IEEE Journal of Biomedical and Health Informatics. 2017; 21(3):838–850. https://doi.org/10.1109/JBHI.2016.2548248

[40] Zhang M, Song W, Zhang J. A secure clinical diagnosis with privacy-preserving multiclass support vector machine in clouds. IEEE Systems Journal. 2022; 16(1):67–78. https://doi.org/10.1109/JSYST.2020.3027758