Practical and Secure SVM Classification for Cloud-Based Remote Clinical Decision Services

Jinwen Liang, Zheng Qin, *Member, IEEE*, Jianbing Ni, *Member, IEEE*, Xiaodong Lin, *Fellow, IEEE*, and Xuemin Shen, *Fellow, IEEE*

Abstract—Support vector machine (SVM) classification techniques have been widely adopted for building clinical decision models. In cloud-based remote clinical decision services, a healthcare center outsources the clinical decision model to a cloud server, which then provides remote clinical decision services to end users. In this article, we propose a practical and secure SVM classification scheme (SSVMC) for cloud-based remote clinical decision services. Specifically, we first extract SVM decision rules from an SVM classifier. Then, we leverage symmetric key encryption to protect the confidentiality of medical data and prevent the cloud service provider from misusing intellectual property of the outsourced clinical model. Finally, we build encrypted indexes to achieve efficient SVM classification. We define a leakage function, formulate a security definition, and provide a simulation-based security proof for SSVMC. The performance analysis demonstrates that SSVMC achieves linear computational complexity when an SVM classifier (a.k.a., the clinical decision model) is pre-trained. The simulations evaluate the impact of several parameters on time costs. The experimental evaluations show the performance differences between SSVMC and several existing schemes in terms of time costs, storage costs, communication costs, and precisions in a real-world clinical dataset, which demonstrate that SSVMC is computationally efficient with high decision accuracy.

Index Terms—Cloud computing, data security, remote clinical decision services, SVM classification, symmetric key encryption

1 Introduction

Driven by the need for high prediction accuracy, easy deployment, and efficient evaluation, Support Vector Machine (SVM) classification techniques have attracted considerable interest in building clinical decision models for diagnosing cancer [2], diabetes [3], heart disease [4], etc. Cloud-based remote clinical decision services are typical applications which provide a remote medical decision for medical features by utilizing pre-trained clinical decision models. There are three entities in cloud-based remote clinical decision services, i.e., a healthcare center, a cloud server, and users [5]. The procedure of cloud-based remote clinical decision services could be described as follows. First, the healthcare center builds a clinical decision model from its' medical dataset by using SVM classification techniques and submits the clinical decision model to a remote cloud server. Second, the third-party cloud server handles massive and frequent clinical decision requests from users and provide clinical decisions based on their medical features and the outsourced SVM decision model.

However, the sensitivity of medical data and the intellectual property protection regulations might hinder the proliferation of cloud-based remote clinical decision services [6], [7], [8]. The user's medical features, such as electrocardiograms, genomics, blood tests, are sensitive data for users. Although it is common to show this information to a doctor at a hospital, users are unwilling to submit their medical features to a remote cloud server, because they may not trust the cloud service provider, which might violate privacy regulations or sell their medical data for monetary reasons [9], [10]. Meanwhile, the corresponding clinical predictions are also sensitive to users. The leakage of clinical predictions may lead to serious issues. For example, such as the health insurance will be increased due to the exposure of bad health status. Moreover, the pre-trained clinical decision model is valuable intellectual properties (IP) for the healthcare center. The healthcare center might worry about IP thefts when it outsources the clinical decision model to a cloud server. For instance, the cloud service provider might share the pre-trained clinical decision model with the competitors of the healthcare center. With such concerns, the confidentiality of both medical data and the clinical decision model must be protected in remote clinical decision services.

To preserve data privacy and enable remote clinical decision services, several secure SVM classification schemes have been proposed [1], [11], [12], [13], [14], [15], [16], [17], [18]. To protect the confidentiality of decision model and medical data while retaining SVM classification functionality, most of those

- *Jinwen Liang is with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China, and also with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada. E-mail: jimmieleung@hnu.edu.cn.*
- *Zheng Qin is with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China. E-mail: zqin@hnu.edu.cn.*
- *Jianbing Ni is with the Department of Electrical and Computer Engineering, Queen's University, Kingston, ON K7L 3N6, Canada. E-mail: jianbing.ni@queensu.ca.*
- *Xiaodong Lin is with the School of Computer Science, University of Guelph, Guelph, ON N1G 2W1, Canada. E-mail: xlin08@uoguelph.ca.*
- *Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada. E-mail: sshen@uwaterloo.ca.*

Manuscript received 30 Jan. 2020; revised 30 June 2020; accepted 23 Aug. 2020. Date of publication 1 Sept. 2020; date of current version 8 Sept. 2021. (Corresponding author: Zheng Qin.) Recommended for acceptance by K. Gaj. Digital Object Identifier no. 10.1109/TC.2020.3020545

schemes are based on time consuming cryptographic techniques, such as secure multi-party computation (MPC) [11], [12], homomorphic encryption (HE) [13], [14], [15], and bilinear pairing [18], which may incur prohibitive computation or communication overheads. Since HE supports arbitrary arithmetic calculations on encrypted data with prohibitive computation requirements, HE-based schemes demand tens of milliseconds for secure SVM classification [14]. Bilinear pairing-based schemes protect the confidentiality of medical data and cost hundreds of milliseconds for secure SVM classification [18]. To improve the computation and communication efficiency, several secure SVM classification schemes adopt techniques such as matrix transformation (MT) [17], order-preserving encryption (OPE) [19], [20], [21], and differential privacy (DP) [22] as essential building blocks. MT-based schemes may leak data distributions of the medical data contents [17]. OPE-based schemes finish SVM classification tasks in microseconds but reveal the numerical orders of medical data contents due to the weak security guarantees of OPE [5]. DP-based schemes [23], [24], [25] achieve high computation efficiency but cannot protect the confidentiality of each individual's medical data [14] and sacrifice the accuracy of clinical decision by adding randomized noises [18]. In summary, there exist three major challenges when designing secure SVM classification schemes for cloud-based remote clinical decision services: (1) Confidentiality: Both medical data and clinical model should be protected against the cloud service provider; (2) Efficiency: Linear computational complexity and microsecond-level execution time should be achieved; and (3) Accuracy: High decision accuracy should be reached.

In this paper, we propose a practical and secure SVM classification scheme (SSVMC) for cloud-based remote clinical decision services to address all the above challenges simultaneously. We first extract decision rules, which are hyper-rectangles with upper boundary and lower boundary at each dimension, from an SVM classifier. Then, the extracted decision rules are considered as multi-dimensional ranges. Accordingly, the SVM classification can be transformed into a function of judging whether a feature vector is located within a multi-dimensional range. Also, we leverage symmetric key encryption to protect the confidentiality of medical data and clinical model. Further, we build encrypted indexes by adopting bloom filter techniques to achieve linear computation complexity for SVM classification. SSVMC achieves high accuracy for real-world remote clinical decision services by extracting decision rules with high precision and choosing a low false positive rate for each bloom filter. The contributions of this paper are three-fold.

- We propose SSVMC for cloud-based remote clinical decision services. Different from previous secure SVM classification schemes, SSVMC leverages symmetric key encryption and pseudo-random functions to protect the confidentiality of medical data and clinical model, which boosts the efficiency of secure SVM classification in terms of time costs. The bloom filter technique is then utilized to construct encrypted indexes for secure SVM classification to achieve linear computation complexity. SSVMC also achieves high decision accuracy in a real-world dataset by choosing appropriate parameters.
- We formulate a security definition and provide a simulation-based security proof for SSVMC. We produce a leakage function $\mathcal{L}$ to formulate the information leakage during the secure SVM classification process. Accordingly, we formulate the adaptive $\mathcal{L}$-security definition for SSVMC. Moreover, we deliver a formal security proof to show that SSVMC captures the adaptive $\mathcal{L}$-security definition, that is, the confidentiality of medical data and clinical model are well protected. The security property achieved in SSVMC is stronger than the OPE-based scheme in [1].
- We conduct extensive performance analyses and evaluations for SSVMC, including performance analysis, simulations, and experiments on real disease diagnosis dataset. The computational complexity shows that SSVMC achieves linear computational complexity once the SVM model is finalized. The simulation results demonstrate the impact of several parameters on time costs. The experiments in real disease diagnosis dataset evaluate the performance differences between SSVMC and several existing schemes in terms of time costs, storage costs, communication costs, and precisions. The evaluation results illustrate that: (1) SSVMC boosts the efficiency in terms of time costs, which achieves microsecond-level execution time for each algorithm in SSVMC; (2) SSVMC requires a low storage costs (less than 20 KB) and a tiny communication costs (less than 400 Bytes) compared with the current cloud storage and wireless network throughput, respectively; and (3) SSVMC achieves 96.19 percent decision accuracy on the Breast-Cancer-Wisconsin dataset.

The remainder of the paper is organized as follows. Section 2 provides the related work. Section 3 describes the system model, threat model, and design goals. Section 4 illustrates preliminaries. Section 5 describes SSVMC in detail. Section 6 formulates the security definition and provides a formal security proof. Section 7 demonstrates the performance analysis and evaluation. Section 8 concludes the paper.

2 Related Work

2.1 Recent Secure SVM Classification Schemes

Support Vector Machine (SVM) classification is a promising technique for making high-stakes decisions in a variety of application fields, such as healthcare [5], finance [14], and transportation [26], [27], [28]. In the field of healthcare, by applying SVM classification, cloud-based remote clinical decision services have shown great potential to provide real-time and high accurate clinical decision services for remote users via mobile devices [29]. However, since the third-party cloud is not fully trusted [30], [31], the healthcare center and the user may worry about the privacy leakage of the clinical decision model and the medical data [6], [7], [8].

To relieve the privacy concerns and enable SVM classification for cloud-based remote clinical decision services, plenty of secure SVM classification schemes have been proposed [1], [11], [12], [13], [14], [15], [16], [17], [18]. Additively homomorphic encryption [32] and fully homomorphic encryption [33], which enable somewhat or fully arbitrary arithmetic computations on encrypted data [34], are straight-forward techniques for designing secure SVM classification schemes by considering SVM classification functionality as a calculation in polynomials. The schemes designed from homomorphic encryption protect both the clinical decision model and the medical data but require heavy computation loads [13], [14], [15]. Moreover, Li *et al.* observed that some homomorphic encryption based schemes may suffer from some soundness and security problems [35]. Multi-party computation also enables secure computations with high security guarantees and incur prohibitive communication overhead. The schemes based on multi-party computation also have some drawbacks such as incur heavy computation load [12] and require trusted zone [11]. In addition, bilinear mapping is a potential technique for designing secure SVM classification schemes with high security properties [18], yet the efficiency of bilinear mapping is still an issue that needs improvement.

To ameliorate both several computation and communication efficiency issues of secure SVM classification, several secure SVM classification schemes are designed from lightweight privacy-preserving building blocks [1], [17], [18]. Matrix transformation is a potential building block for secure SVM classification protocol, but cannot achieve provable security properties [17]. By transforming an SVM classifier to several geographic range rules [36], order-preserving encryption (OPE) [19], [20], [21] can also be used as building blocks for secure SVM classification schemes. Yet the schemes based on OPE leak the numerical order information of sensitive data [1]. Differential privacy is a possible technique for devising secure SVM classification, but it reduces the decision accuracy due to the noise adding on the SVM classifier [24], [25].

2.2 Summary of Changes

This paper is an extended version of the conference paper [1]. We briefly summarize the main differences as follows.

1) We propose a different secure SVM classification scheme (SSVMC) for cloud-based remote clinical decision services. The conference version in [1] is an OPE-based scheme, which will inevitably leak the order information of data contents. Different from the scheme in [1], SSVMC is derived from bloom filters, symmetric key encryption, pseudo-random functions, and pseudo-random permutations, which enhances the security property and ensures the computational efficiency in terms of time costs.
2) We formulate the adaptively $\mathcal{L}$-secure definition for SSVMC, which is strictly stronger than the security property achieved in [1]. The security property achieved in [1] is indistinguishability under frequency analysis ordered chosen-plaintext attack (IND-FAOCPA) [20], which is the ideal security property for OPE with frequency hiding but inevitably leak the order information of data contents. We formulate the leakage function $\mathcal{L}$, define the adaptively $\mathcal{L}$-secure definition, and provide a simulation-based security proof for SSVMC.
3) We conduct more performance analysis and evaluations to demonstrate the computation efficiency and precision of SSVMC. Specifically, we add computation cost analysis, simulations, and experimental evaluations in a real-world dataset. The analysis and the evaluation results demonstrate that SSVMC is computationally efficient with high classification accuracy.

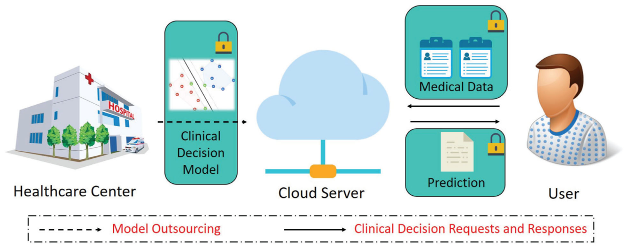


Fig. 1. The system model for secure cloud-based remote clinical decision services.

3 Models and Design Goals

3.1 System and Threat Model

We show the system model of cloud-based remote clinical decision services in Fig. 1, which involves three entities, i.e., a healthcare center ($\mathcal{HC}$), a cloud server ($\mathcal{CS}$), and a user ($\mathcal{U}$). $\mathcal{CS}$ is an *honest-but-curious* entity, and both $\mathcal{HC}$ and $\mathcal{U}$ are considered as honest entities. Thus, the clinical decision model and the medical data should be protected against $\mathcal{CS}$. The procedure of secure cloud-based remote clinical decision services can be identified as follows.

1) *Healthcare Center* ($\mathcal{HC}$). $\mathcal{HC}$ applies the SVM classification technique to produce a clinical decision model (i.e., SVM classifier) from sensitive medical data such as electronic health records (EHRs). Then, $\mathcal{HC}$ encrypts the SVM decision model and outsources the encrypted clinical decision model to $\mathcal{CS}$. Finally, $\mathcal{HC}$ sends several security parameters to $\mathcal{U}$.
2) *Cloud Server* ($\mathcal{CS}$). After receiving the encrypted clinical decision model, $\mathcal{CS}$ provides remote clinical decision services to $\mathcal{U}$. Specifically, $\mathcal{CS}$ applies the encrypted clinical decision model to $\mathcal{U}$'s medical feature vectors and produces the corresponding encrypted clinical prediction to $\mathcal{U}$.
3) *User* ($\mathcal{U}$). $\mathcal{U}$ uses the security parameters from $\mathcal{HC}$ to encrypt his/her medical features and submits his/her encrypted medical features to $\mathcal{CS}$. Then, $\mathcal{U}$ uses the security parameters from $\mathcal{HC}$ to decrypt the encrypted clinical prediction from $\mathcal{CS}$.

3.2 Design Goals

We aim to design a practical and secure SVM classification scheme for cloud-based remote clinical decision services. The proposed scheme should achieve the following properties.

1) *Confidentiality*. To protect $\mathcal{U}$'s data privacy and $\mathcal{HC}$'s intellectual property, the $\mathcal{U}$'s medical data, the clinical prediction (i.e., *data confidentiality*), and $\mathcal{HC}$'s clinical decision model (i.e., *model confidentiality*) should be protected.
2) *Efficiency*. To provide real-time remote clinical decision services, linear computational complexity and microsecond-level execution time should be achieved.
3) *Accuracy*. To provide accurate remote clinical decision services, high clinical precision should be reached.

4 Preliminaries

4.1 Symmetric Key Encryption

Symmetric key encryption (SKE) denotes the encryption scheme whose encryption key is the same as the decryption key. Any SKE such as Advanced Encryption Standard (AES) is probabilistic methods, in which produced ciphertext is indistinguishable from a random value [5]. We consider the pseudo-randomness against chosen-plaintext attacks (PCPA) notion for SKE [5]. SKE is defined as follows.

Definition 1. *Symmetric Key Encryption. Symmetric key encryption involves three algorithms, i.e.,* SKE = (SKE.Gen, SKE.Enc, SKE.Dec). *Let k be a private key. κ is a security parameter, Msg is a message, $\widehat{Msg}$ is a ciphertext.* SKE *could be illustrated as follows:*

- $k \leftarrow$ SKE.Gen(1^κ).
- $\widehat{Msg} \leftarrow$ SKE.Enc(k, Msg).
- $Msg \leftarrow$ SKE.Dec$(k, \widehat{Msg})$.

To protect data confidentiality, AES is utilized to implement PCPA-secure SKE.

4.2 Pseudo-Random Functions

A pseudo-random function denotes a function that produces pseudo-random outputs which are computationally indistinguishable from random values [37]. The definition is as follows.

Definition 2. *Pseudo-random Functions. $Prf : \{0,1\}^\kappa \times \{0,1\}^t \rightarrow \{0,1\}^l$ is an keyed function. We say Prf is a pseudo-random function if for all probabilistic polynomial-time adversary $\mathcal{A}$, there exists a negligible function $negl$ such that*

$$\left|\Pr[\mathcal{A}^{Prf_k(\cdot)}(1^t) = 1] - \Pr[\mathcal{A}^{Rnd_t(\cdot)}(1^t) = 1]\right| \leq negl(t),$$

where the key $k \leftarrow \{0,1\}^\kappa$ is a randomly chosen κ-bit string and Rnd_t is randomly selected from the set of functions mapping t-bit strings to l-bit strings.

The standard Hash-based Message Authentication Code (HMAC) [38] is utilized to implement the pseudo-random functions.

4.3 Pseudo-Random Permutations

A pseudo-random permutation denotes a keyed bijection whose output cannot be computationally distinguished from a permutation which is randomly chosen from the set of all permutations on the function's domain [38]. The definition is as follows.

Definition 3. *Pseudo-random Permutations. Let $Prp : \{0,1\}^\kappa \times \{0,1\}^t \rightarrow \{0,1\}^t$ be an efficient, keyed permutation. We say Prp is a pseudo-random permutation if for all probabilistic polynomial-time adversary $\mathcal{A}$, there exists a negligible function $negl$ such that*

$$\left|\Pr[\mathcal{A}^{Prp_k(\cdot)}(1^t) = 1] - \Pr[\mathcal{A}^{Rnd_t(\cdot)}(1^t) = 1]\right| \leq negl(t),$$

where the key $k \leftarrow \{0,1\}^\kappa$ is a randomly chosen κ-bit string and Rnd_t is selected uniformly at random from the set of all functions permutating t-bit strings to t-bit strings.

4.4 Bloom Filters

Bloom filters are efficient data structures for indicating whether a data element is either definitely not in a set with several data elements or possibly in the set [39]. A bloom filter can be implemented by a vector with m bits and k hash functions, i.e., $BF = [b_1, b_2, \ldots, b_m]$, and $h_1, h_2, \ldots, h_k$. Generally, the bloom filter data structure contains three operations, i.e., *initialization*, *adding*, and *detection*. The initialization operation initializes all m bit elements to $'0'$. The adding operation adds a data item to the bloom filter by calculating k hash results for the data item and setting all these k bits of BF to $'1'$. For instance, when adding an element $'a'$ to BF, the adding operation sets $BF[h_i(a)] = 1$, where $i = 1, 2, \ldots, k$. The detection operation indicates whether a data element is definitely not in BF or possibly in BF (i.e., return false or true) by calculating k hash results for the data element and evaluating whether these k positions in BF are all $'1'$. For instance, when detecting whether element $'c'$ in BF, the detection operation calculates $BF[h_i(c)]$, where $i = 1, 2, \ldots, k$, and judging whether all these k positions in BF are all $'1'$, which is

$$\bigwedge_{i=1}^{k} BF[h_i(c)] = BF[h_1(c)] \wedge \cdots \wedge BF[h_k(c)] \stackrel{?}{=} 1.$$

False Positives (*FP*). False positive probability affects the correctness of indicating whether a data element is stored in a bloom filter. The false positive probability of a bloom filter BF could be calculated as

$$P_{fp}(BF) = \left(1 - \left(1 - \frac{1}{m}\right)^{kd}\right)^k \approx \left(1 - e^{-kd/m}\right)^k, \tag{1}$$

where d is the number of data elements in BF, m is the length of BF, and k is the number of hash functions. The minimized $P_{fp}(BF)$ is $2^{-k} \approx 0.6185^{m/d}$, when $k = \ln 2 \times (m/d)$ [39]. Further analysis of the false positive could be found in [40].

4.5 Support Vector Machine Classification and Rule Extraction

Support Vector Machine (SVM) is a robust classification technique with high predictive accuracy [18]. The SVM classification technique makes predictions for data by utilizing a separating hyper-plane (a.k.a, SVM classifier). Generally, an SVM classifier is shown in the following equation, i.e., Eq. (2)

$$f(x) = sign\left(\sum_{j=1}^{m} \alpha_j y_j K(\boldsymbol{s}_j, \boldsymbol{x}) + b\right), \tag{2}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is a n dimension feature vector, $y_j \in \{-1, +1\}$ is the corresponding prediction of $\boldsymbol{s}_j$, α_j is the Lagrange multipliers, $\boldsymbol{s}_j$ is the jth support vector, m is the number of support vectors, b is the bias, and $K(\boldsymbol{s}_j, \boldsymbol{x})$ is the kernel function [41].

To express the SVM classifier, Fu *et al.* [36] extract SVM rules from the boundary of hyper-rectangles, which are transformed from an SVM classifier. The SVM rule extraction scheme contains three phases: expression, initialization, and optimization. We provide an example with two predictions in Fig. 2 to describe the SVM rule extraction scheme in [36]. The example can be spread to a case with multiple predictions by utilizing one-versus-all policy [41]. The example

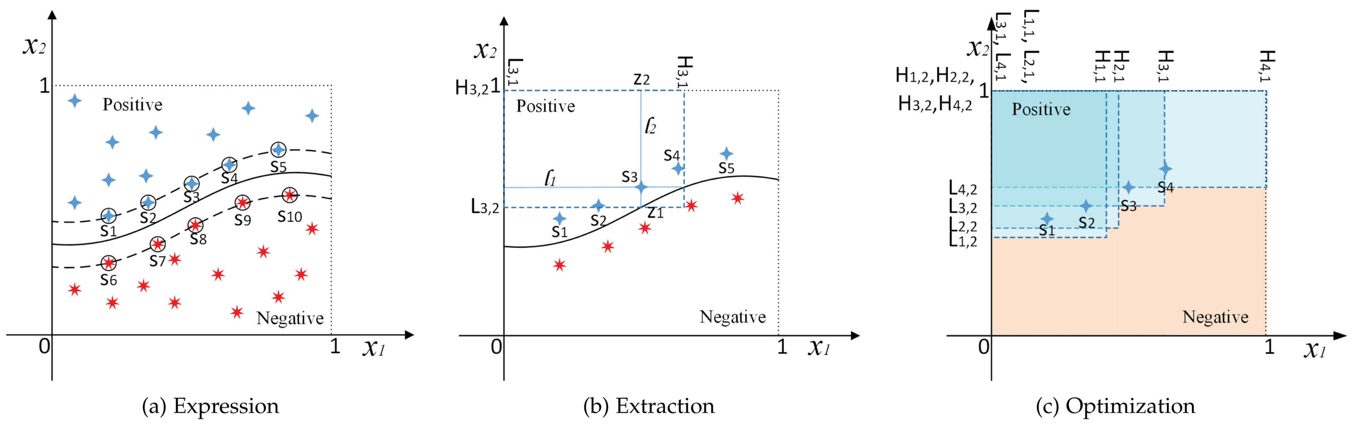


Fig. 2. A simple example of extracting hyper-rectangles from the SVM classifier.

in Fig. 2 contains two predictions, i.e., positive (the blue one) and negative (the red one). The input feature vector $\boldsymbol{x}$ contains two dimensions, i.e., $\boldsymbol{x} = \{x_1, x_2\}$. The value of $\boldsymbol{x}$ in each dimension is normalized to the interval [0,1] by using somewhat normalization techniques.

Expression. In this phase, an SVM classifier could be expressed by a hyper-plane the largest margin and several support vectors [41]. In Fig. 2a, support vectors are the feature vectors with black rings, i.e., five support vectors with positive (blue) prediction ($\boldsymbol{s}_1, \ldots, \boldsymbol{s}_5$) and five support vectors with negative (red) prediction ($\boldsymbol{s}_6, \ldots, \boldsymbol{s}_{10}$). The hyper-plane is the black solid curve in Fig. 2a.

Extraction. In this phase, several hyper-rectangles are initially extracted by utilizing support vectors with positive (blue) prediction (blue), i.e., $\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \boldsymbol{s}_4, \boldsymbol{s}_5$. We provide an extraction example for $\boldsymbol{s}_3$ in Fig. 2b. Crossing the support vector $\boldsymbol{s}_3$, line l_1 and line l_2 could be finalized, which are parallel to dimension x_1 and x_2, respectively. Then, l_2 crosses the separating hyper-plane at z_1, and the boundary of dimension x_2 at z_2. After that, the lower boundary and the upper boundary of the hyper-rectangle produced by $\boldsymbol{s}_3$ at dimension x_2 are $L_{3,2}$ and $H_{3,2}$, respectively. Similarly, the lower boundary and the upper boundary of $\boldsymbol{s}_3$ at dimension x_1 are finalized at $L_{3,1}$ and $H_{3,1}$ respectively. As a result, the boundary of a hyper-rectangle produced from $\boldsymbol{s}_3$ can be initially extracted. By reusing the above method, five hyper-rectangles can be extracted by using the support vectors with positive (blue) prediction.

Optimization. In this phase, the boundary of each hyper-rectangle will be tuned, and the redundant hyper-rectangles will be pruned. First, based on the initial hyper-rectangles extracted from support vectors with positive prediction, whenever a feature vector $\boldsymbol{x}$ with negative (red) prediction falls into the region of a hyper-rectangle, the boundary of the hyper-rectangle will be tuned to exclude the outliers. Second, whenever a hyper-rectangle area is a subset of the region covered by other certain hyper-rectangles, the hyper-rectangle will be pruned. After the tuning and pruning phase, the extracted hyper-rectangles could be found in Fig. 2c. The blue region could be considered as the boundary of positive prediction, and the red zone could be viewed as the boundary of negative prediction.

After transforming the SVM classifier to several hyper-rectangles, 5 decision rules could be finalized in Fig. 3 with the region of the extracted hyper-rectangles in Fig. 2c. As a result, the SVM classifier could be expressed by using the region of hyper-rectangles extracted from support vectors. Please kindly note that the number of extracted rules cannot be estimated because the optimization phase may prune some redundant rules and thus reduce the number of rules. We leverage Fu *et al.*'s method [36] as the building block of SVM classifier expression.

5 The Proposed SSVMC

5.1 Definitions

Let $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$ be an n dimensional feature vector. *SVM* denotes an SVM classifier trained from medical data. $\boldsymbol{R} = \{R_1, R_2, \ldots, R_t\}$ denotes t rules extracted from *SVM*. For each rule $R_i \in \boldsymbol{R}$, $R_i = \{R_{i,1}, R_{i,2}, \ldots, R_{i,j}, \ldots, R_{i,n}\}$. Each $R_{i,j} \in \boldsymbol{R}$ contains two values, i.e., $R_{i,j}.low$ and $R_{i,j}.up$, which denote the lower boundary and the upper boundary of the rule R_i in dimension j, respectively. Given $R_{i,j}.low$ and $R_{i,j}.up$, we enumerate a set of all possible values that are inside the interval $[R_{i,j}.low, R_{i,j}.up]$. Note that it is trivial to enumerate values inside an interval in plaintexts [39]. $EI(R_{i,j})$ denotes all values enumerated in the interval $[R_{i,j}.low, R_{i,j}.up]$. For the ease of description, all data in this paper are positive integers, because other types of data can be transformed to positive integers in plaintexts. We use $t \times n$ vectors as bloom filters $\boldsymbol{BF} = \{BF_{1,1}, \ldots, BF_{i,j}, \ldots, BF_{t,n}\}$ to store all intervals. Namely, $BF_{i,j}$ stores values in $EI(R_{i,j})$, where $i = 1, 2, \ldots, t$ and $j = 1, 2, \ldots, n$. $m_{i,j}$ denotes the length of $BF_{i,j}$. *FP* denotes the false positive rate for all bloom filters $BF_{i,j}$, which is determined by k and $m_{i,j}$. Let

Extracted Rules From SVM Classifier

Rule 1: IF ($L_{1,1} \le x_1 \le H_{1,1}$) And ($L_{1,2} \le x_2 \le H_{1,2}$)
Then *the prediction of* $\boldsymbol{x}$ *is positive* (*Blue*);
Rule 2: IF ($L_{2,1} \le x_1 \le H_{2,1}$) And ($L_{2,2} \le x_2 \le H_{2,2}$)
Then *the prediction of* $\boldsymbol{x}$ *is positive* (*Blue*);
Rule 3: IF ($L_{3,1} \le x_1 \le H_{3,1}$) And ($L_{3,2} \le x_2 \le H_{3,2}$)
Then *the prediction of* $\boldsymbol{x}$ *is positive* (*Blue*);
Rule 4: IF ($L_{4,1} \le x_1 \le H_{4,1}$) And ($L_{4,2} \le x_2 \le H_{4,2}$)
Then *the prediction of* $\boldsymbol{x}$ *is positive* (*Blue*);
Default
Then *the prediction of* $\boldsymbol{x}$ *is negative* (*Red*).

Fig. 3. Decision rules extracted from hyper-rectangles.

M be the sum of all $m_{i,j}$, i.e.,

$$M = \sum_{i=1}^{t} \sum_{j=1}^{n} m_{i,j}.$$

$\boldsymbol{y} = \{y_1, y_2, \ldots, y_t, y_0\}$ denotes the clinical prediction. Let y_i be the prediction of R_i, where $i = 1, \ldots, t$, and y_0 be the prediction of the default rule.

κ denotes the security parameter. Pseudo-random functions (HMAC) is utilized to implement hash functions for $BF_{i,j}$, where $i = 1, 2, \ldots, t$ and $j = 1, 2, \ldots, n$. $\boldsymbol{K} = \{K^0_{1,1}, \ldots, K^p_{i,j}, \ldots, K^k_{t,n}\}$ are $t * n * (k+1)$ HMAC secret keys for $\boldsymbol{BF}$, where $i = 1, \ldots, t$, $j = 1, \ldots, n$, $p = 0, \ldots, k$. $K^0_{i,j}, \ldots, K^k_{i,j}$ are $k+1$ HMAC secret keys for $BF_{i,j}$. h_0 and h_1 are the M bit pseudo-random permutation and the $\log(t+1)$ bit pseudo-random permutation, respectively, where K_0 and K_1 are two secret keys for h_0 and h_1, respectively. A is the random permutation of an M-bit array, which is the concatenation of all bloom filters. Let $A[l]$ be the lth element in the array A, where $l = 0, 1, \ldots, M-1$. $\boldsymbol{K^*} = \{K^0, \ldots, K^i, \ldots, K^t\}$ are $t+1$ AES symmetric keys. Let $\widehat{y_i}$ be the encrypted y_i. The index T stores the random permutated addresses of $\widehat{y_0}, \widehat{y_1}, \ldots, \widehat{y_t}$. Let $T[l]$ be the lth element of index T, where $l = 0, 1, \ldots, t$. $\boldsymbol{TK} = \{TK_0, TK_1, \ldots, TK_t\}$ are $t+1$ tokens for achieving secure SVM classification. TK_i denotes the token for R_i, where $i = 1, 2, \ldots, t$. TK_0 denotes the token for default rule. For each tokens TK_i, there are $n * k$ different location addresses $L_{i,j,p}$ in A and an address $addr(i)$ in T, where $i = 1, 2, \ldots, t$.

All notations are shown in Table 1 and SSVMC is described as follows.

Definition 4. *Secure Support Vector Machine Classification* (SSVMC). SSVMC *contains four polynomial-time algorithms, i.e.,* SSVMC= (Init, ClfEnc, TokenGen, Eva).

- Init $(\kappa) \rightarrow \boldsymbol{K}, \boldsymbol{K^*}, K_0, K_1, h_0, h_1$. *The initialization algorithm is run by $\mathcal{HC}$. Based on the security parameter κ, $\mathcal{HC}$ produces $\boldsymbol{K}$, $\boldsymbol{K^*}$, K_0, K_1, h_0, h_1 and sends these parameters to authorized $\mathcal{U}$.*
- ClfEnc $(\boldsymbol{K}, K_0, K_1, h_0, h_1, \boldsymbol{K^*}, \boldsymbol{R}) \rightarrow A, T$. *The classifier encryption algorithm is run by $\mathcal{HC}$. First, $\mathcal{HC}$ calculates $EI(R_{i,j})$, where $i = 1, \ldots, t$ and $j = 1, \ldots, n$. Then, $\mathcal{HC}$ adds all $EI(R_{i,j})$ to A by adopting the idea of bloom filters. After that, $\mathcal{HC}$ encrypts y_i and stores the encrypted y_i at T, where $i = 0, \ldots, t$. Finally, $\mathcal{HC}$ outsources A and T to $\mathcal{CS}$.*
- TokenGen $(\boldsymbol{K}, K_0, K_1, h_0, h_1, \boldsymbol{x}) \rightarrow \boldsymbol{TK}$. *The token generation algorithm is run by $\mathcal{U}$. When $\mathcal{U}$ makes a clinical decision for his/her medical features $\boldsymbol{x}$, he/she produces a set of tokens $\boldsymbol{TK}$ for $\boldsymbol{x}$, and outsources the $\boldsymbol{TK}$ to $\mathcal{CS}$.*
- Eva $(\boldsymbol{TK}, \boldsymbol{K^*}, A, T) \rightarrow y_i$. *The evaluation algorithm is an interactive algorithm run by $\mathcal{CS}$ and $\mathcal{U}$. After receiving $\boldsymbol{TK}$ from $\mathcal{U}$, $\mathcal{CS}$ produces a clinical decision by using A and T and returns the encrypted prediction to $\mathcal{U}$. Then $\mathcal{U}$ decrypts the encrypted prediction and receives y_i as the evaluation result for $\boldsymbol{x}$, where $i = 0, 1, \ldots, t$.*

5.2 Detailed Descriptions

In Fig. 4, we briefly describe the main idea of SSVMC by illustrating the work-flow of the classifier encryption, token generation, and evaluation processes.

TABLE 1
Important Notations

Notations	Descriptions
SVM	The SVM classifier trained from medical data.
n	The number of dimensions of feature vectors.
$\boldsymbol{x}$	$\boldsymbol{x} = \{x_1, \ldots, x_i, \ldots, x_n\}$ is an n dimensional feature vector. x_i denotes the ith feature of $\boldsymbol{x}$.
t	The number of rules.
$\boldsymbol{R}$	$\boldsymbol{R} = \{R_1, \ldots, R_i, \ldots, R_t\}$ are t rules extracted from an SVM classifier.
R_i	$R_i = \{R_{i,1}, \ldots, R_{i,j}, \ldots, R_{i,n}$ is the ith rule in $\boldsymbol{R}$. $R_{i,j}$ denotes the lower boundary and the upper boundary of R_i at jth dimension, where $R_{i,j} = (R_{i,j}.low, R_{i,j}.up)$.
$EI(R_{i,j})$	All values enumerated in the interval $[R_{i,j}.low, R_{i,j}.up]$.
$\boldsymbol{y}$	$\boldsymbol{y} = \{y_1, \ldots, y_i, \ldots, y_t, y_0\}$ are $t+1$ predictions for an SVM classifier.
y_i	The prediction of R_i, where $i = 1, \ldots, t$.
y_0	The prediction of the default rule.
$\boldsymbol{BF}$	$\boldsymbol{BF} = \{BF_{1,1}, \ldots, BF_{i,j}, \ldots, BF_{t,n}\}$ denotes a $t \times n$ vectors to store all intervals, i.e., $BF_{i,j}$ stores $EI(R_{i,j})$.
$m_{i,j}$	The length of $BF_{i,j}$.
M	The sum length of $BF_{i,j}$.
FP	The false positive rate for $\boldsymbol{BF}$.
κ	The security parameter.
$\boldsymbol{K}$	$\boldsymbol{K} = \{K^0_{1,1}, \ldots, K^p_{i,j}, \ldots, K^k_{t,n}\}$ denotes $t * n * (k+1)$ HMAC keys for $\boldsymbol{BF}$. $K^p_{i,j}$ denotes HMAC keys for $BF_{i,j}$.
h_0	The M bit pseudo-random permutation.
h_1	The $\log(t+1)$ bit pseudo-random permutation.
K_0	The secret key for h_0.
K_1	The secret key for h_1.
A	The M bit array to store $\boldsymbol{BF}$.
$\boldsymbol{K^*}$	$\boldsymbol{K^*} = \{K^0, \ldots, K^t\}$ denotes $t+1$ AES symmetric keys.
$\widehat{y_i}$	The encrypted prediction for R_i, i.e., the encrypted y_i.
T	The table with $t+1$ elements to store encrypted y_i.
$\boldsymbol{TK}$	$\boldsymbol{TK} = \{TK_0, TK_1, \ldots, TK_t\}$ denotes $t+1$ tokens.
$L_{i,j,p}$	The location indicator of x_j for R_i in A, where $p = 1, 2.., k$.
$addr(i)$	The prediction address for TK_i in T.

In the classifier encryption process, since several decision rules could be extracted from an SVM classifier [36], $\mathcal{HC}$ uses the decision rules which are tuples contain R_i and y_i to denote the clinical decision model. $\mathcal{HC}$ then constructs two encrypted indexes A and T for the decision rules $\boldsymbol{R}$ and the corresponding prediction $\boldsymbol{y}$, respectively. In Fig. 4, $\boldsymbol{R}$ contains t different rules, which are the boundaries of hyper-rectangles extracted from the SVM classifier. There are n intervals in rule R_i, where $R_i \in \boldsymbol{R}$. Each $R_{i,j} \in R_i$ denotes the boundary of R_i in dimension j. $\mathcal{HC}$ enumerates all values that located in the interval $[R_{i,j}.low, R_{i,j}.up]$ and adds $EI(R_{i,j})$ to the bloom filter $BF_{i,j}$. For example, for value v that located in the interval $[R_{i,j}.low, R_{i,j}.up]$, $\mathcal{HC}$ sets the location $\text{HMAC}(K^0_{i,j}, \text{HMAC}(K^p_{i,j}, v)) \bmod m_{i,j}$ in $BF_{i,j}$ to be '1', where $p = 1, \ldots, k$. Then $\mathcal{HC}$ concatenates all $BF_{i,j}$ as an M-bit array A and randomly permutates all values in A by using h_0. Meanwhile, there are t clinical predictions that correspond to $\boldsymbol{R}$ and 1 clinical prediction corresponds to default in $\boldsymbol{y}$. $\mathcal{HC}$ applies AES to encrypt each clinical prediction. Then, $\mathcal{HC}$ uses h_1 to permutate the encrypted predictions

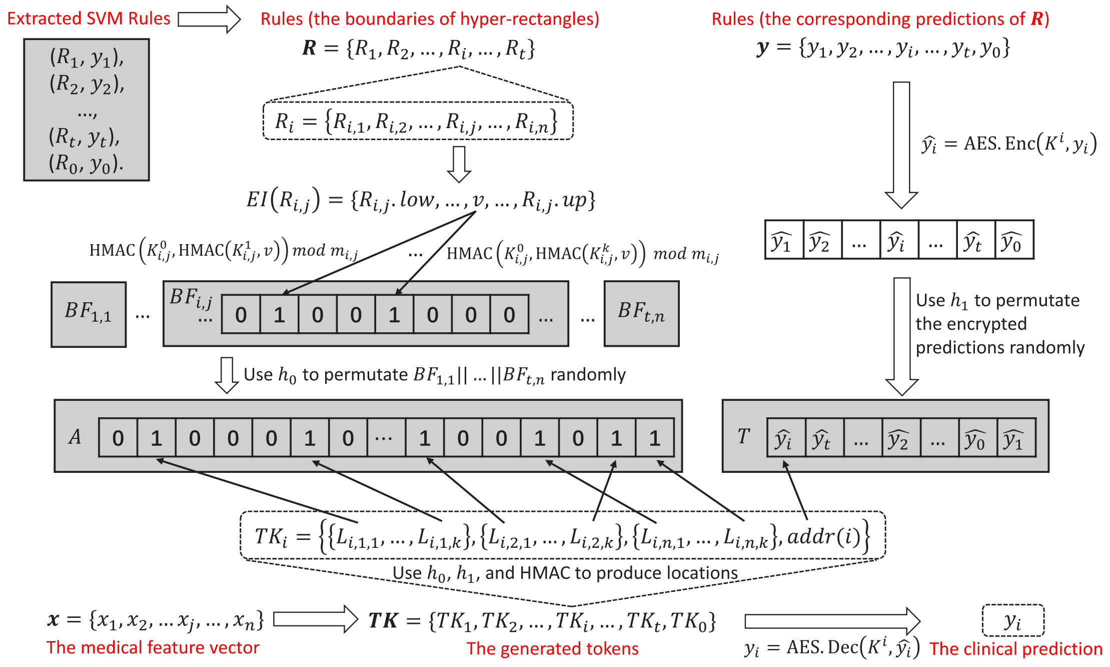

Extracted SVM Rules
(R_1, y_1), (R_2, y_2), ..., (R_t, y_t), (R_0, y_0).
Rules (the boundaries of hyper-rectangles)
$\boldsymbol{R} = \{R_1, R_2, \ldots, R_i, \ldots, R_t\}$
$R_i = \{R_{i,1}, R_{i,2}, \ldots, R_{i,j}, \ldots, R_{i,n}\}$
$EI(R_{i,j}) = \{R_{i,j}.low, \ldots, v, \ldots, R_{i,j}.up\}$
$\mathrm{HMAC}(K^0_{i,j}, \mathrm{HMAC}(K^1_{i,j}, v)) \bmod m_{i,j}$
$\mathrm{HMAC}(K^0_{i,j}, \mathrm{HMAC}(K^k_{i,j}, v)) \bmod m_{i,j}$
$BF_{1,1}$ $BF_{i,j}$ $BF_{t,n}$
Use h_0 to permutate $BF_{1,1}||\ldots||BF_{t,n}$ randomly
Rules (the corresponding predictions of $\boldsymbol{R}$)
$\boldsymbol{y} = \{y_1, y_2, \ldots, y_i, \ldots, y_t, y_0\}$
$\widehat{y_i} = \mathrm{AES.Enc}(K^i, y_i)$
Use h_1 to permutate the encrypted predictions randomly
A T
$TK_i = \{\{L_{i,1,1}, \ldots, L_{i,1,k}\}, \{L_{i,2,1}, \ldots, L_{i,2,k}\}, \{L_{i,n,1}, \ldots, L_{i,n,k}\}, addr(i)\}$
Use h_0, h_1, and HMAC to produce locations
$\boldsymbol{x} = \{x_1, x_2, \ldots x_j, \ldots, x_n\}$
The medical feature vector
$\boldsymbol{TK} = \{TK_1, TK_2, \ldots, TK_i, \ldots, TK_t, TK_0\}$
The generated tokens
$y_i = \mathrm{AES.Dec}(K^i, \widehat{y_i})$
y_i
The clinical prediction

Fig. 4. The classifier encryption, token generation, and evaluation processes of SSVMC.

randomly and stores all encrypted predictions in the table T. Finally, the encrypted index A and T are produced.

In the token generation process, $\mathcal{U}$ produces $\boldsymbol{TK}$ for the medical vector $\boldsymbol{x}$ by using h_0, h_1, and HMAC. As shown in Fig. 4, $\boldsymbol{TK}$ contains $t+1$ tokens, i.e., $TK_0, TK_1, TK_2, \ldots, TK_t$. TK_i is a token that corresponds to R_i, which contains $n*k$ location indicators $L_{i,j,p}$ and an address $addr(i)$, where $i = 1, \ldots, t$. In TK_i, $L_{i,j,p}$ indicates the pth location address of $R_{i,j}$ in the encrypted index A, where $p = 1, \ldots, k$. $addr(i)$ indicates the address of encrypted prediction $\widehat{y_i}$ in the encrypted index T. TK_0 is the token for the default rule R_0, which contains $addr(0)$ in T.

In the evaluation process, $\mathcal{CS}$ uses $\boldsymbol{TK}$ to search the corresponding clinical prediction y_i from A and T. As shown Fig. 4, the bitwise-AND for the value of TK_i in A equals '1', i.e., $\bigwedge_{j=1}^{n}\bigwedge_{p=1}^{k} A[L_{i,j,p}] = 1$, which denotes that $\boldsymbol{x}$ satisfies R_i. Then, $\mathcal{CS}$ returns $\widehat{y_i} = T[addr(i)]$ to $\mathcal{U}$. Finally, $\mathcal{U}$ uses AES to decrypt $\widehat{y_i}$ and obtains the clinical prediction y_i for $\boldsymbol{x}$. Note that when the bitwise-AND result for all rules are equals to '0', i.e., $\bigvee_{i=1}^{t}\bigwedge_{j=1}^{n}\bigwedge_{p=1}^{k} A[L_{i,j,p}] = 0$, $\mathcal{CS}$ returns $T[addr(0)]$ as the default encrypted prediction for $\boldsymbol{x}$, and $\mathcal{U}$ applies AES to obtain the clinical prediction y_0.

We provide a detailed description of SSVMC in Fig. 5.

6 Security Definitions and Analyses

To analyze the security property of SSVMC, we formulate a leakage function $\mathcal{L}$ to indicate the leaked information during the SVM classification process. Then, we provide a simulation-based definition to formulate the adaptively $\mathcal{L}$-secure definition. After that, we give a formal security proof to demonstrate that with all but negligible probability, the adversary cannot distinguish the output of the real experiment from that of the simulated experiment. Finally, we compare the security properties achieved in SSVMC and that in the schemes in [1].

6.1 Leakage Functions and Security Definition

We first define a leakage function $\mathcal{L}$ to describe the information leakage revealed from the secure SVM classification processing history [39], [42], [43]. The inputs of SSVMC are the SVM rules $\boldsymbol{R}$ and the feature vectors $\boldsymbol{x}$. $\mathcal{L}(\boldsymbol{R}, \boldsymbol{x})$ is defined as follows.

Definition 5. $\mathcal{L}(\boldsymbol{R}, \boldsymbol{x})$. *The leakage function contains size pattern, search pattern, and access pattern.*

- Size pattern. *The size pattern contains t, $m_{i,j}$, M, and $Len(\mathtt{AES})$. t is the number of rules in $\boldsymbol{R}$, which also indicates the length of the index T, because T contains $t+1$ elements. $m_{i,j}$ is the length of each $BF_{i,j}$. M denotes the length of the array A. $Len(\mathtt{AES})$ denotes the length of ciphertexts produced by* AES.
- Access pattern. *$\alpha(\cdot)$ denotes the access pattern. The access pattern contains $\alpha(\boldsymbol{x})$, $\alpha(addr(i))_{R_i\in\boldsymbol{R}}$, and $\alpha(L_{i,j,p})_{R_i\in\boldsymbol{R},\, x_j\in\boldsymbol{x}}$. $\alpha(\boldsymbol{x})$ is the SVM prediction y_i that corresponds to $\boldsymbol{x}$. $\alpha(addr(i))_{R_i\in\boldsymbol{R}}$ is the linkage between rule R_i (or token TK_i) and the corresponding prediction address for R_i in T. $L_{i,j,p}$ are k locations for R_i and x_j in $BF_{i,j}$, which is pseudo-randomly permutated in A, where $p = 1, 2, \ldots, k$. $\alpha(L_{i,j,p})_{R_i\in\boldsymbol{R},\, x_j\in\boldsymbol{x}}$ is the linkage between these k locations to the corresponding prediction address $addr(i)$ in T.*
- Search pattern. *$\beta(\cdot)$ denotes the search pattern. $\beta(\boldsymbol{x})$ denotes the difference between two input feature vectors. $\beta(addr(i))$ is the difference between two addresses in index T. $\beta(L_{i,j,p})$ denotes the difference between two locations in A.*

Then, we formulate the leakage function as follows:

$$\mathcal{L}(\boldsymbol{R}, \boldsymbol{x}) = <\ t, M, m_{i,j}, Len(\mathtt{AES}), \alpha(\boldsymbol{x}), \beta(\boldsymbol{x}), \\ (\alpha(addr(i)), \beta(addr(i)))_{R_i\in\boldsymbol{R}}, \\ (\alpha(L_{i,j,p}), \beta(L_{i,j,p}))_{R_i\in\boldsymbol{R},\, x_j\in\boldsymbol{x}} > .$$

Secure Support Vector Machine Classification (SSVMC)

★ **Initialization**: SSVMC.Init $(\kappa) \rightarrow \boldsymbol{K}, \boldsymbol{K}^*, K_0, K_1, h_0, h_1$.

1: $\mathcal{HC}$: $\mathcal{HC}$ chooses a security parameter κ, randomly produces $t * n * (k+1)$ keys $\boldsymbol{K}$ for `HMAC`, i.e.,

$$K^p_{i,j} \xleftarrow{\$} \{0,1\}^\kappa,$$

where $i = 1, \ldots, t$, $j = 1, \ldots, n$, $p = 0, \ldots, k$. Then, $\mathcal{HC}$ produces $t+1$ symmetric keys $\boldsymbol{K}^*$ for `AES`, i.e.,

$$K^i \leftarrow \texttt{AES.KeyGen}(1^\kappa),$$

where $i = 0, \ldots, t$. After that, $\mathcal{HC}$ randomly produces two pseudo-random permutation h_0 and h_1, where

$$h_0 : \{0,1\}^\kappa \times \{0,1\}^M \rightarrow \{0,1\}^M, \quad h_1 : \{0,1\}^\kappa \times \{0,1\}^{\log(t+1)} \rightarrow \{0,1\}^{\log(t+1)}.$$

$\mathcal{HC}$ randomly produces two keys K_0 and K_1 for h_0 and h_1, respectively. That is,

$$K_0 \xleftarrow{\$} \{0,1\}^\kappa, \quad K_1 \xleftarrow{\$} \{0,1\}^\kappa.$$

2: $\mathcal{HC} \rightarrow \mathcal{U}$: $\mathcal{HC}$ sends $\boldsymbol{K}$, $\boldsymbol{K}^*$, K_0, K_1, h_0, and h_1 to authorized $\mathcal{U}$.

★ **Classifier Encryption**: SSVMC.ClfEnc ($\boldsymbol{K}$, K_0, K_1, h_0, h_1, $\boldsymbol{K}^*$, $\boldsymbol{R}$) $\rightarrow A, T$.

1: $\mathcal{HC}$: For each rule $R_i = \{R_{i,1}, R_{i,2}, \ldots, R_{i,n}\} \in \boldsymbol{R}$, $\mathcal{HC}$ enumerates all values in $R_{i,j}$ and produces $EI(R_{i,j})$, where $j = 1, 2, \ldots, n$.

2: $\mathcal{HC}$: For each value $v \in EI(R_{i,j})$, $\mathcal{HC}$ sets

$$A\left[h_0\left(K_0, \left(\sum_{ii=1}^{i-1}\sum_{jj=1}^{n} m_{ii,jj} + \sum_{jj=1}^{j-1} m_{i,jj} + \texttt{HMAC}\left(K^0_{i,j}, \texttt{HMAC}\left(K^p_{i,j}, v\right)\right) \mod m_{i,j}\right)\right)\right] = 1,$$

where $p = 1, 2, \ldots, k$.

3: $\mathcal{HC}$: For each clinical prediction y_i, $\mathcal{HC}$ calculates

$$addr(i) = h_1(K_1, i), \quad \widehat{y_i} = \texttt{AES.Enc}(K^i, y_i),$$

and sets

$$T[addr(i)] = \widehat{y_i},$$

where $i = 0, 1, \ldots, t$.

4: $\mathcal{HC} \rightarrow \mathcal{CS}$: $\mathcal{HC}$ outsources A and T to $\mathcal{CS}$.

★ **Token Generation**: SSVMC.TokenGen ($\boldsymbol{K}$, K_0, K_1, h_0, h_1, $\boldsymbol{x}$) $\rightarrow \boldsymbol{TK}$.

1: $\mathcal{U}$: When $\mathcal{U}$ makes a clinical prediction for his/her medical features $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$, he/she produces

$$L_{i,j,p} = h_0\left(K_0, \left(\sum_{ii=1}^{i-1}\sum_{jj=1}^{n} m_{ii,jj} + \sum_{jj=1}^{j-1} m_{i,jj} + \texttt{HMAC}\left(K^0_{i,j}, \texttt{HMAC}(K^p_{i,j}, x_j)\right) \mod m_{i,j}\right)\right),$$

where $i = 1, 2, \ldots, t$, $j = 1, 2, \ldots, n$, $p = 1, 2, \ldots, k$. Then, $\mathcal{U}$ calculates $addr(i) = h_1(K_1, i)$, where $i = 0, 1, \ldots, t$.

2: $\mathcal{U}$: $\mathcal{U}$ produces $(t+1)$ tokens for $\boldsymbol{x}$, i.e., $\boldsymbol{TK} = \{TK_0, TK_1, \ldots, TK_t\}$. For TK_i, $\mathcal{U}$ produces

$$TK_i = \{\{L_{i,1,1}, \ldots, L_{i,1,k}\}, \{L_{i,2,1}, \ldots, L_{i,2,k}\}, \ldots, \{L_{i,n,1}, \ldots, L_{i,n,k}\}, addr(i)\},$$

where $i = 1, 2, \ldots, t$. For TK_0, $\mathcal{U}$ produces $TK_0 = \{addr(0)\}$.

3: $\mathcal{U} \rightarrow \mathcal{CS}$: $\mathcal{U}$ submits $\boldsymbol{TK}$ to $\mathcal{CS}$.

★ **Evaluation**: SSVMC.Eva ($\boldsymbol{TK}$, $\boldsymbol{K}^*$, A, T) $\rightarrow y_i$.

1: $\mathcal{CS}$: $\mathcal{CS}$ receives A and T from $\mathcal{HC}$.

2: $\mathcal{CS}$: After receiving $\boldsymbol{TK}$ from $\mathcal{U}$, $\mathcal{CS}$ calculates

$$Flag_i = \bigwedge_{j=1}^{n}\bigwedge_{p=1}^{k} A[L_{i,j,p}],$$

where $i = 1, 2, \ldots, t$.

3: $\mathcal{CS} \rightarrow \mathcal{U}$: If $Flag_i = 1$, then $\mathcal{CS}$ searches $T[addr(i)]$ and obtains $\widehat{y_i}$, where i is a value located in the interval $[0, t]$. Else if $\bigvee_{i=1}^{t} Flag_i = 0$, $\mathcal{CS}$ searches $T[addr(0)]$ and obtains $\widehat{y_0}$. Then, $\mathcal{CS}$ returns $\widehat{y_i}$ to $\mathcal{U}$.

4: $\mathcal{U}$: After receiving $\widehat{y_i}$, $\mathcal{U}$ calculates

$$y_i = \texttt{AES.Dec}(K_i, \widehat{y_i}),$$

where i is a value located in the interval $[0, t]$. Finally, y_i is the corresponding clinical prediction for $\boldsymbol{x}$.

Fig. 5. Details of SSVMC for cloud-based remote clinical decision services.

With the leakage function $\mathcal{L}$, the adaptive $\mathcal{L}$-security definition is formulated as follows.

Definition 6. *Adaptive $\mathcal{L}$-security. Let $\Pi = (\mathsf{Init}, \mathsf{ClfEnc}, \mathsf{TokenGen}, \mathsf{Eva})$ be a secure SVM classification scheme. Let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_q)$ be an adversary such that $q \in \mathbb{N}$ and $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_q)$ be a simulator. Let $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^q$ be feature vectors for q SVM classification requests produced by $\mathcal{A}$. We define two experiments $\mathbf{Real}_{\Pi}^{\mathcal{A}}(1^\kappa)$ and $\mathbf{Sim}_{\mathcal{L},\mathcal{S}}^{\mathcal{A}}(1^\kappa)$ as follows.*

$\mathbf{Real}_{\Pi}^{\mathcal{A}}(1^\kappa)$: At round 0, the challenger runs $\mathsf{Init}(\kappa)$ to generate $\boldsymbol{K}$, $\boldsymbol{K}^*$, K_0, K_1, h_0, and h_1. $\mathcal{A}_0$ produces SVM rules $\boldsymbol{R}$. The challenger runs $\mathsf{ClfEnc}(\boldsymbol{K}, \boldsymbol{K}^*, K_0, K_1, h_0, h_1, \boldsymbol{R})$ and sends the output A and T to $\mathcal{A}$. Then, $\mathcal{A}$ makes q classification requests, at round r ($1 \leq r \leq q$): $\mathcal{A}_r$ produces $\boldsymbol{x}^r$ based on the previous requests. The challenger calculates $\boldsymbol{TK}^r$ from $\mathsf{TokenGen}(\boldsymbol{K}, \boldsymbol{K}^*, K_0, K_1, h_0, h_1, \boldsymbol{x})$ and sends $\boldsymbol{TK}^r$ to $\mathcal{A}_r$. After q round interactions, $\mathcal{A}$ produces a bit as the output.

$\mathbf{Sim}_{\mathcal{L},\mathcal{S}}^{\mathcal{A}}(1^\kappa)$: At round 0, $\mathcal{A}_0$ produces SVM rules $\boldsymbol{R}$. Based on the leakage function $\mathcal{L}(\boldsymbol{R}, \boldsymbol{x})$, $\mathcal{S}_0$ produces two arrays A^* and T^* and sends A^*, T^* to $\mathcal{A}$. Then, $\mathcal{A}$ makes q classification requests, at round r ($1 \leq r \leq q$): $\mathcal{A}_r$ produces $\boldsymbol{x}^r$. Based on $\mathcal{L}(\boldsymbol{R}, \boldsymbol{x})$, $\mathcal{S}_r$ produces $t * n * k$ random locations (i.e., $L_{i,j,p}^*$) in A and $t+1$ addresses in T associated with all $L_{i,j,p}^*$ as an appropriate token $\boldsymbol{TK}^{r*}$ and sends $\boldsymbol{TK}^{r*}$ to $\mathcal{A}_r$. After q round interactions, $\mathcal{A}$ produces a bit as the output.

We say that Π is adaptively $\mathcal{L}$-secure if for all polynomial size adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_q)$, there exists a simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_q)$ and a negligible function $negl(\kappa)$ such that

$$\left| Pr\left[\mathbf{Real}_{\Pi}^{\mathcal{A}}(1^\kappa) = 1\right] - Pr\left[\mathbf{Sim}_{\mathcal{L},\mathcal{S}}^{\mathcal{A}}(1^\kappa) = 1\right] \right| \leq negl(\kappa).$$

6.2 Security Proof

Theorem 1. *SSVMC is adaptively $\mathcal{L}$-secure if h_0 and h_1 are pseudo-random permutations, $\mathtt{HMAC}$ is pseudo-random function, and $\mathtt{AES}$ is PCPA-secure symmetric key encryption.*

Proof. We produce a polynomial-size simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_q)$ such that for a polynomial-size adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_q)$, the output of $\mathbf{Real}_{\Pi}^{\mathcal{A}}(1^\kappa)$ and $\mathbf{Sim}_{\mathcal{L},\mathcal{S}}^{\mathcal{A}}(1^\kappa)$ are computationally indistinguishable. Consider $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_q)$ that adaptively produces A^*, T^*, and $\boldsymbol{TK}^{r*}$ as follows.

$\mathcal{S}_0$: With the leakage function $\mathcal{L}(\boldsymbol{R}, \boldsymbol{x})$, $\mathcal{S}_0$ obtains M and $m_{i,j}$, where $i = 1, 2, \ldots, t$ and $j = 1, 2, \ldots, n$. To simulate A^*, $\mathcal{S}_0$ first constructs an array A^* with M bits and initializes all bits in A^* to '0'. Then, for each $m_{i,j}$, $\mathcal{S}_0$ chooses $m_{i,j}$ unmarked bits in A^* uniformly and randomly, and marks these $m_{i,j}$ bits as $BF_{i,j}^*$ locally. Afterwards, $\mathcal{S}_0$ randomly chooses k different locations in $BF_{i,j}^*$ and sets the values of these locations in A^* to '1'. After setting $t * n * k$ random locations in A^* to '1', the simulated A^* is produced. With the leakage function $\mathcal{L}(\boldsymbol{R}, \boldsymbol{x})$, $\mathcal{S}_0$ obtains t and $Len(\mathtt{AES})$. To produce T^*, $\mathcal{S}_0$ constructs an index T^* with $t+1$ elements, and sets each element l in T^* to be $T[l] \xleftarrow{\$} \{0,1\}^{Len(\mathtt{AES})}$. Finally, $\mathcal{S}_0$ produces A^* and T^* and sends A^* and T^* to $\mathcal{A}_0$.

With all but negligible probability, $\mathcal{A}$ cannot obtain K_0 and $K_{i,j}^0$. Since it is hard to distinguish between the pseudo-random locations generated by $h_0(K_0, *)$ and $\mathtt{HMAC}$ and that of randomly selected locations, A^* is indistinguishable from A. With all but negligible probability, $\mathcal{A}$ cannot obtain $\boldsymbol{K}^*$. Thus, the encrypted prediction $\widehat{y_i}$ in T is computationally indistinguishable from random string with the same length in T^* if $\mathtt{AES}$ achieves PCPA security property. Since T^* is an index with $t+1$ element such that each element stores a random string with the same length as the output of $\mathtt{AES}$, T^* is computationally indistinguishable from T.

$\mathcal{S}_r$: For $1 \leq r \leq q$: With $\beta(\boldsymbol{x})$ from $\mathcal{L}(\boldsymbol{R}, \boldsymbol{x})$, $\mathcal{S}_r$ checks whether the feature vector $\boldsymbol{x}^r = \{x_1^r, x_2^r, \ldots, x_n^r\}$ has appeared before. There are three possibilities. First, $\boldsymbol{x}^r$ doesn't appear before, $\mathcal{S}_r$ produces $t * n * k$ identifiers $L_{i,j,p}^{r*}$ in A^* and $t+1$ addresses $addr(i)^*$ that associates with identifiers in A^*. For each rule R_i and each feature x_j^r, $\mathcal{S}_r$ locates $BF_{i,j}^*$ in A^* and randomly chooses k different locations in $BF_{i,j}^*$ as $L_{i,j,1}^{r*}, L_{i,j,2}^{r*}, \ldots, L_{i,j,k}^{r*}$. When $r = 1$, $\mathcal{S}_r$ selects a permutation uniformly at random that permutates a string with $t+1$ values, i.e., $Rnd_{t+1}(\cdot)$, and sets $addr(i)^* = Rnd_{t+1}(i)$ for each token TK_i. When $r \neq 1$, with $(\alpha(addr(i)))_{R_i \in \boldsymbol{R}}$ from $\mathcal{L}(\boldsymbol{R}, \boldsymbol{x})$, $\mathcal{S}_r$ produces $addr(i)^*$ that associates with these locations and $TK_i^{r*} = \{L_{i,j,1}^{r*}, \ldots, L_{i,j,k}^{r*}, addr(i)^*\}$, where $i = 1, 2, \ldots, t$ and $j = 1, 2, \ldots, n$. Finally, $\mathcal{S}_r$ produces $TK_0^{r*} = addr(0)^*$ and outputs $\boldsymbol{TK}^{r*} = \{TK_1^{r*}, \ldots, TK_t^{r*}, TK_0^{r*}\}$ to $\mathcal{A}_r$. Second, $\boldsymbol{x}^r$ has totally appeared before. Namely, $\exists\ \boldsymbol{x}^u$, such that $\forall\ x_j^r \in \boldsymbol{x}^r$ and $x_j^r = x_j^u$, where $1 \leq u < r$. Then, $\mathcal{S}_r$ returns $\boldsymbol{TK}^{r*} = \boldsymbol{TK}^{u*}$ as the appropriate token for $\boldsymbol{x}^r$ to $\mathcal{A}_r$ by using $\alpha(\boldsymbol{x})$, $\beta(\boldsymbol{x})$, $(\alpha(addr(i)), \beta(addr(i)))_{R_i \in \boldsymbol{R}}$, and $\left(\alpha(L_{i,j,p}), \beta(L_{i,j,p})\right)_{R_i \in \boldsymbol{R},\ x_j \in \boldsymbol{x}}$ from $\mathcal{L}(\boldsymbol{R}, \boldsymbol{x})$. Third, $\boldsymbol{x}^r$ has partially appeared before. Namely, $\exists\ x_j^r \in \boldsymbol{x}^r$, such that $x_j^r = x_j^u$, where $1 \leq u < r$. For x_j^r that doesn't appear before, $\mathcal{S}_r$ produces $t * k$ identifier $L_{i,j,p}^{r*}$ in A^* by using the method in the first possibility. For x_j^r that has appeared before, $\mathcal{S}_r$ produces $t * k$ identifier $L_{i,j,p}^{r*}$ in A^* by using $\beta(L_{i,j,p})_{R_i \in \boldsymbol{R},\ x_j \in \boldsymbol{x}}$. Given $(\alpha(addr(i)), \beta(addr(i)))_{R_i \in \boldsymbol{R}}$, $\mathcal{S}_r$ produces $t+1$ identifiers for TK_i^{r*} in T^*. Finally, $\mathcal{S}_r$ submits $\boldsymbol{TK}^{r*}$ to $\mathcal{A}_r$.

With all but negligible probability, $\mathcal{A}$ cannot obtain K_1. Thus, $addr(i)^{r*}$ is indistinguishable from $addr(i)^r$ under the assumption that it is hard to distinguish between the output of $h_1(K_1, *)$ and that of randomly selected permutation. Similarly, $\mathcal{A}$ cannot obtain $\boldsymbol{K}$, K_0, and h_0, the location set $\{L_{i,j,1}^{r*}, L_{i,j,2}^{r*}, \ldots, L_{i,j,k}^{r*}\}$ for R_i and x_j^r is computationally indistinguishable from $\{L_{i,j,1}^{r}, L_{i,j,2}^{r}, \ldots, L_{i,j,k}^{r}\}$ under the assumption that it is hard to distinguish between pseudo-random locations generated by $h_0(K_0, *)$ and $\mathtt{HMAC}$ and that of randomly selected locations, because $\{L_{i,j,1}^{r*}, L_{i,j,2}^{r*}, \ldots, L_{i,j,k}^{r*}\}$ are randomly selected from $BF_{i,j}^*$ in A^*. Thus, $\boldsymbol{TK}^{r*}$ is indistinguishable from $\boldsymbol{TK}^r$. Meanwhile, for features that partially repeated or totally repeated from the previous classification requests, $\boldsymbol{TK}^{r*}$ or TK_j^{r*} is the same as $\boldsymbol{TK}^{u*}$ or TK_j^{u*}, where $1 \leq u < r$. Since $\boldsymbol{TK}^{u*}$ and TK_j^{u*} are indistinguishable from $\boldsymbol{TK}^u$ and TK_j^u, the repeated $\boldsymbol{TK}^{r*}$ and TK_j^{r*} are indistinguishable from $\boldsymbol{TK}^r$ and TK_j^r. Therefore, with all but negligible probability, $\boldsymbol{TK}^{r*}$ is indistinguishable from $\boldsymbol{TK}^r$.

Therefore, $\mathcal{A}$ cannot distinguish the output of $\mathbf{Sim}_{\mathcal{L},\mathcal{S}}^{\mathcal{A}}(1^\kappa)$ from $\mathbf{Real}_{\Pi}^{\mathcal{A}}(1^\kappa)$. □

TABLE 2
Computational Costs

Algorithm	Computational Costs
Init	$t * n * (k+1) * C'_{\texttt{HMAC}} + (t+1) * C'_{\texttt{AES}} + 2 * C'_{prp}$
ClfEnc	$t * n * d * k * (2 * C_{\texttt{HMAC}} + C_{prp}) + (t+1) * (C_{\texttt{AES}} + C_{prp})$
TokenGen	$2 * t * n * k * C_{\texttt{HMAC}} + (t+1) * C_{prp}$
Eva	$t * n * k * C_{and} + C_{\texttt{AES}}$

In summary, both the clinical decision model and the medical data are well protected. Compared with the scheme in [1], which adopts IND-FAOCPA secure order-preserving encryption, SSVMC significantly improves the security properties by protecting the order information of the model and the data. To build low-latency, real-time cloud-based remote clinical decision services, SSVMC is constructed based on pseudo-random bloom filters, which leak the size pattern, the search pattern, and the access pattern. The information leakage also exists in some searchable symmetric encryption schemes [37], [43], [44] and privacy-preserving data classification schemes [1], [5]. To enhance the security with ensured search and access pattern protection, techniques such as homomorphic encryption [13], [14], [15], multi-party computation [12] have been applied for designing secure SVM classification schemes. However, such approaches inevitably consume prohibitive computational or communication load. Also, re-encryption is a potential way to enable stronger search and access pattern protection for SSVMC by periodically reset the leakage functions [5], [43].

7 Performance Analyses and Evaluations

7.1 Performance Analysis

To evaluate the performance, we provide computational cost analysis for SSVMC with respect to several parameters. Let t, n, d, and k be the number of rules, the number of dimensions, the number of enumerated values in $BF_{i,j}$, and the number of hash functions in $BF_{i,j}$, respectively. Let $C'_{\texttt{HMAC}}$ and $C_{\texttt{HMAC}}$ be the costs of HMAC key generation and HMAC computation, respectively. Let $C'_{\texttt{AES}}$ and $C_{\texttt{AES}}$ be the costs of AES key generation, AES computation (encryption or decryption), respectively. Let C'_{prp} and C_{prp} be the costs of pseudo-random permutation key generation and computation, respectively. Let C_{and} be the cost of bitwise-AND computation. Table 2 demonstrates the computation costs of each algorithm in SSVMC.

Different from many secure SVM classification schemes that constructed from time-consuming cryptographic techniques such as homomorphic encryption [14], multi-party computation [12], and bilinear mapping [18], SSVMC brings lower computational overhead, as it applies lightweight privacy-preserving techniques, including symmetric key encryption, pseudo-random functions and permutations. Meanwhile, once the SVM rules R are extracted and the false positive rate of $BF_{i,j}$ is finalized, As shown in Table 2, the computational complexity of SSVMC is linear to the number of input data.

7.2 Experiment Settings

We conduct both simulation experiments and performance evaluations in real dataset to show the performance of SSVMC. SSVMC is implemented in `C++` code based on `OpenSSL`.[1] The experiments are conducted on a 64-bit VMware Workstation (running Ubuntu 18.04) with an Intel Core i7-8850H CPU with 2.60 GHz and 8 GB RAM. AES-CBC-256 and HMAC-512 are utilized to implement `AES` and `HMAC`, respectively.

False Positives. Let m, d, k be the length of bloom filter, the number of elements in bloom filter, and the number of hash function in bloom filter, respectively. To calculate the false positive rate (*FP*) in each $BF_{i,j}$, k and m/d in Eq. (1) are set as follows. (1) $FP = 10^{-2}$: $k = 5$ and $m/d = 10$; (2) $FP = 10^{-3}$: $k = 7$ and $m/d = 15$; (3) $FP = 10^{-4}$: $k = 8$ and $m/d = 21$; (4) $FP = 10^{-5}$: $k = 11$ and $m/d = 25$.

Simulation Setting. The number of data items in each bloom filter is assumed to be the same. Different SVM rules are randomly produced to satisfy the simulation settings. We evaluate the following setting for each algorithm in SSVMC. (1) Number of rules is from 10 to 50, and 30 by default; (2) Number of dimensions for each feature vector is from 10 to 50, and 30 by default; (3) Number of data items in each bloom filter is from 10 to 50, and 30 by default.

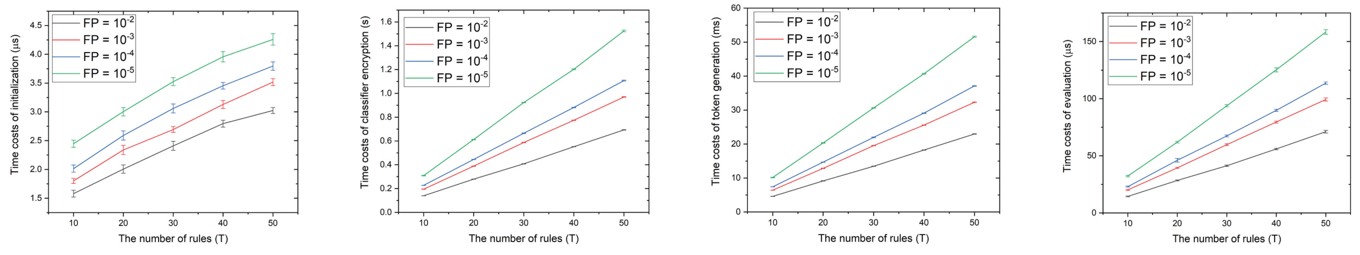
Datasets. The Breast-Cancer-Wisconsin[2] dataset is utilized to train the clinical decision model using SVM classification. There are 683 non-missing data records in this dataset. For each record, there are 9 discrete attributes, whose value is located in the interval [1,10]. In this dataset, 458 records' predictions are benign (65.5 percent) and 241 records' predictions are malignant (34.5 percent).

Extracted Rules. After training a clinical decision model by using SVM classification in the Breast-Cancer-Wisconsin dataset, we extract 4 rules from the clinical decision model. Let A_i be the attribute i in the dataset, and $[a, b]$ be a discrete interval for each attribute. All values in the Breast-Cancer-Wisconsin dataset are discrete and ranges in a domain [1,10]. The SVM rules are shown as follow.

R_1: $A_1 \in [1,7]$, $A_2 \in [1,9]$, $A_3 \in [1,9]$, $A_4 \in [1,7]$, $A_5 \in [1,6]$, $A_6 \in [1,6]$, $A_7 \in [1,9]$, $A_8 \in [1,6]$, $A_9 \in [1,6]$, Benign;
R_2: $A_1 \in [1,7]$, $A_2 \in [1,9]$, $A_3 \in [1,9]$, $A_4 \in [1,7]$, $A_5 \in [1,6]$, $A_6 \in [1,6]$, $A_7 \in [1,8]$, $A_8 \in [1,6]$, $A_9 \in [1,7]$, Benign;
R_3: $A_1 \in [1,7]$, $A_2 \in [1,9]$, $A_3 \in [1,9]$, $A_4 \in [1,7]$, $A_5 \in [1,6]$, $A_6 \in [1,7]$, $A_7 \in [1,6]$, $A_8 \in [1,7]$, $A_9 \in [1,2]$, Benign;
R_0: Default, Malignant.

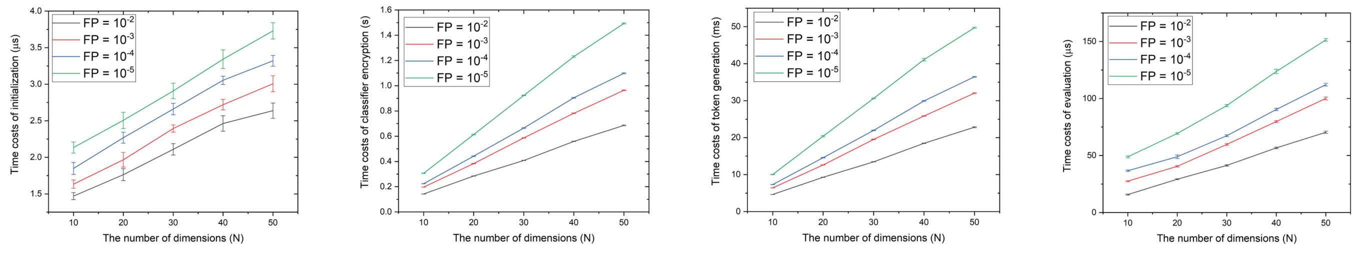
Baselines. We evaluate the performance advantages of SSVMC compared with several existing schemes about time costs and precisions. Three secure SVM classification schemes are taken into account as baselines. (1) *BPTG15* is designed based on additive homomorphic encryption. We implement *BPTG15* by using `libhcs`[3] and `GMP`.[4] Since the tested dataset is two-class dataset, we only need to implement the private dot product protocol for implementing *BPTG15*. Thus, *BPTG15* is implemented by Paillier additive encryption. (2) *ZLLL16* is designed based on bilinear paring. We use `PBC`[5] library to implement *ZLLL16*. (3) *LQNLS19* is designed based on order-preserving encryption. We use

1. [Online]. Available: https://www.openssl.org/
2. [Online]. Available: http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/
3. [Online]. Available: https://github.com/tiehuis/libhcs
4. [Online]. Available: https://gmplib.org/
5. [Online]. Available: https://crypto.stanford.edu/pbc/



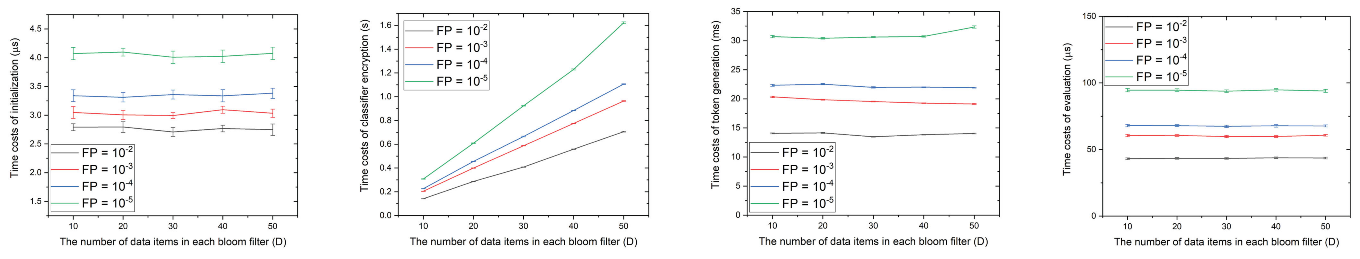
(a) Initiation (b) Classifier Encryption (c) Token Generation (d) Evaluation

Fig. 6. Impact of t on time costs for SSVMC.



(a) Initiation (b) Classifier Encryption (c) Token Generation (d) Evaluation

Fig. 7. Impact of n on time costs for SSVMC.



(a) Initiation (b) Classifier Encryption (c) Token Generation (d) Evaluation

Fig. 8. Impact of d on time costs for SSVMC.

IND-FAOCPA secure [20] order-preserving encryption to implement *LQNLS19*.

7.3 Simulation Results

In the simulation, we make 500 experimental runs and show the average time costs. We also compute the 95 percent confidence intervals as error bars.

Impact of t on Time Costs. We evaluate the impact of the number of rules on time costs for SSVMC and show the simulation results in Fig. 6. Fig. 6a illustrates the time cost of initiation algorithm in SSVMC grows linearly when t increases. Fig. 6b demonstrates the time cost of classifier encryption algorithm in SSVMC grows linearly when t increases. Fig. 6c shows the time cost of token generation algorithm in SSVMC grows linearly when t increases. Fig. 6d describes the time cost of evaluation algorithm in SSVMC grows linearly when t increases. The simulation results demonstrate the time cost of SSVMC grows linearly when t grows linearly.

Impact of n on Time Costs. We evaluate the impact of the number of dimensions on time costs for SSVMC and show the simulation results in Fig. 7. Fig. 7a illustrates the time cost of initiation algorithm in SSVMC grows linearly when t increases. Fig. 7b demonstrates the time cost of classifier encryption algorithm in SSVMC grows linearly when t increases. Fig. 7c shows the time cost of token generation algorithm in SSVMC grows linearly when t increases. Fig. 7d describes the time cost of evaluation algorithm in SSVMC grows linearly when t increases. The simulation results demonstrate the time cost of SSVMC grows linearly when n grows linearly.

Impact of d on Time Costs. We evaluate the impact of the number of data items in each bloom filter $BF_{i,j}$ on time costs for SSVMC and show the simulation results in Fig. 8. Fig. 8a illustrates the time cost of initiation algorithm in SSVMC keeps constant when d increases. Fig. 8b demonstrates the time cost of classifier encryption algorithm in SSVMC grows linearly when d increases. Fig. 8c shows the time cost of token generation algorithm in SSVMC keeps constant when d increases. Fig. 8d describes the time cost of evaluation algorithm in SSVMC keeps constant when t increases. The simulation results demonstrate the time cost of classifier encryption algorithm in SSVMC grows linearly and the time cost of initiation, token generation, and evaluation algorithms keeps constant when d grows linearly.

7.4 Performance Evaluations in Real Dataset

We evaluate the performance advantages of SSVMC compared with *BPTG15*, *ZLLL16*, and *LQNLS19* in the

TABLE 3
Performance Differences in the Breast Cancer Wisconsin Dataset

Schemes	Time Costs (μs)				Precision					Stor. and Comm. Costs (Bytes)		
	Init	ClfEnc	TokenGen	Eva	TP	FN	FP	TN	ACC	Storage	Token	Result
BPTG15 [14]	38009	N/A	75704.02	3073.29	433	11	7	232	97.36%	N/A	2304	256
ZLLL16 [18]	647818	N/A	3405.17	556197.39	433	11	7	232	97.36%	N/A	576	576
LQNLS19 [1]	0.27	1.81	1.74	0.18	428	16	10	229	96.19%	1856	576	32
SSVMC (10^{-2})	5	2724	223.58	1.45	428	16	10	229	96.19%	6540	127	32
SSVMC (10^{-3})	7	4266	308.69	2.03	428	16	10	229	96.19%	9746	189	32
SSVMC (10^{-4})	8	4729	350.68	2.36	428	16	10	229	96.19%	13594	221	32
SSVMC (10^{-5})	10	6960	489.30	3.22	428	16	10	229	96.19%	16159	321	32

Breast-Cancer-Wisconsin dataset. For SSVMC, we evaluate the performance of SSVMC when choosing different false positives, i.e., $FP = 10^{-2}$, 10^{-3}, 10^{-4}, and 10^{-5}.

We show the average time cost of initialization, classifier encryption, token generation, and evaluation for SSVMC, *BPTG15*, *ZLLL16*, and *LQNLS19* in Table 3. Compared with *BPTG15* and *ZLLL16* designed from asymmetric encryption, SSVMC boosts efficiency in terms of time costs. Since *BPTG15* and *ZLLL16* are designed from homomorphic encryption and bilinear pairing, respectively, they do not encrypt the SVM classifier separately. Thus, the classifier encryption process is not applicable to these two schemes. By leveraging order-preserving encryption to protect the data privacy and model privacy, *LQNLS19* inevitably faces to the leakage of the order information as well as several patterns, and achieves better efficiency than SSVMC in terms of average time cost. We also evaluate SSVMC with different false positive rates for each bloom filter $BF_{i,j}$. The results show that SSVMC requires microseconds for each algorithm, which demonstrate that SSVMC is efficient for cloud-based remote clinical decision services.

The precision of SVM classification is also an important factor for providing accurate remote clinical decision services. To evaluate the precision, we count the number of true positive (TP), false negative (FN), false positive (FP), and true negative (TN) and calculate the accuracy (ACC) for SSVMC, *BPTG15*, *ZLLL16*, and *LQNLS19*, Fu *et al.*'s [36] method is utilized to extract SVM rules from the SVM classifier trained from the Breast-Cancer-Wisconsin dataset, as shown in Section 7.2. The evaluation results are shown in Table 3. The asymmetric encryption based schemes such as *BPTG15* and *ZLLL16* encrypt all parameters in the SVM classifier, and therefore the accuracy of secure SVM classification keeps the same as the original SVM classification in the plaintext form. The evaluation results demonstrate that the classification accuracy of both *BPTG15* and *ZLLL16* achieves 97.36 percent. Different from the asymmetric encryption based schemes, both *LQNLS19* and SSVMC are designed for the extracted SVM rules. The accuracy of *LQNLS19* is the same as the extracted rules in Section 7.2, which achieves 96.19 percent. SSVMC is constructed by using bloom filter, which may lead to accuracy loss due to the false positive rate for each $BF_{i,j}$. The experimental results show that there is no accuracy lost in SSVMC when *FP* is less than 10^{-2}. SSVMC achieves 96.19 percent accuracy in the Breast-Cancer-Wisconsin dataset. Therefore, SSVMC achieves high accuracy for real-world remote clinical decision services.

As shown in Table 3, we evaluate the storage and communication costs of SSVMC, *BPTG15*, *ZLLL16*, and *LQNLS19*. The column "storage" denotes the storage costs of the outsourced encrypted SVM classifier. The column "token" denotes the bandwidth costs of submitting secure SVM classification requests. The column "result" denotes the bandwidth costs of receiving encrypted predictions. Please kindly note that *ZLLL16* contains two-round communication, and thus we show the total bandwidth of requests and responses in the "token" column and the "result" column, respectively. Both *BPTG15* and *ZLLL16* are designed in a server-client system model, and thus these schemes don't need to outsource the encrypted SVM classifier to a cloud server. Table 3 demonstrates that the storage costs of SSVMC for different chosen false positives (i.e., $FP = 10^{-2}$, 10^{-3}, 10^{-4}, and 10^{-5}) are less than 20 KB in the tested dataset, which are low storage costs for $\mathcal{CS}$. Table 3 also shows that the communication costs of SSVMC are lower than that of *BPTG15*, *ZLLL16*, and *LQNLS19*. Furthermore, the communication costs of SSVMC for submitting the secure SVM classification requests and receiving secure SVM classification results are less than 400 Bytes in the tested dataset, which is a tiny cost compared with the current wireless network throughput.

8 Conclusion

In this paper, we have proposed SSVMC to protect the confidentiality of clinical model and medical data for cloud-based remote clinical decision services using SVM classification. Compared with existing secure SVM classification schemes, SSVMC extremely boosts the efficiency of secure SVM classification in terms of time costs. Specifically, we have defined the leakage function $\mathcal{L}$ to evaluate the information leakage during the secure SVM classification process. Accordingly, we have formulated the adaptive $\mathcal{L}$-security definition and provided a formal security proof to demonstrate that SSVMC captures the adaptive $\mathcal{L}$-security definition. Security analysis has shown that SSVMC achieves stronger security properties than the scheme in [1]. The performance analysis and evaluation results show that SSVMC achieves linear computational complexity for SVM classification, finishes SVM classification tasks in microseconds in the Breast-Cancer-Wisconsin dataset, and achieves 96.19 percent clinical decision accuracy in the Breast-Cancer-Wisconsin dataset. Consequently, SSVMC addresses the confidentiality, efficiency, and accuracy challenges simultaneously, and is a practical solution for secure cloud-based remote clinical decision services. For the future work, we will focus on the challenge that the leakage of size pattern, search pattern, and access pattern exists in some secure data classification schemes, and further enhance the security properties of secure SVM scheme against malicious adversaries.

Acknowledgment

This work was partially supported by the National Key R&D Projects under Grant 2018YFB0704000 and Grant 2017YFB0902904, in part by National Natural Science Foundation of China under Grant 61772191, in part by the China Scholarship Council under Grant 201806130132, in part by the Science and Technology Key Projects of Hunan Province under Grant 2018TP1009, Grant 2015TP1004, Grant 2019GK2082, in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada, and the Science and Technology Key Projects of Changsha City under Grant kq1804008 and Grant kq1902051. Part of this research work was presented in IEEE International Conference of Communications (ICC 2019) [1].

References

[1] J. Liang, Z. Qin, J. Ni, X. Lin, and X. Shen, "Efficient and privacy-preserving outsourced SVM classification in public cloud," in *Proc. IEEE Int. Conf. Commun.*, 2019, pp. 1–6.

[2] S. Huang, N. Cai, P. P. Pacheco, S. Narrandes, Y. Wang, and W. Xu,"Applications of support vector machine (SVM) learning in cancer genomics," *Cancer Genomics-Proteomics*, vol. 15, no. 1, pp. 41–51, 2018.

[3] K. Polat, S. Güneş, andA. Arslan,"A cascade learning system for classification of diabetes disease: Generalized discriminant analysis and least square support vector machine," *Expert Syst. Appl.*, vol. 34, no. 1, pp. 482–487, 2008.

[4] T. Mythili, D. Mukherji, N. Padalia, andA. Naidu,"A heart disease prediction model using SVM-decision trees-logistic regression (SDL)," *Int. J. Comput. Appl.*, vol. 68, no. 16, pp. 11–15, 2013.

[5] J. Liang, Z. Qin, S. Xiao, L. Ou, and X. Lin, "Efficient and secure decision tree classification for cloud-assisted online diagnosis services," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2019.2922958.

[6] T. Rabesandratana, "E.U. privacy protection bill would hamper research, scientists warn," *Science*, vol. 343, no. 6174, pp. 959–960, 2014.

[7] C. Zhang, L. Zhu, C. Xu, and R. Lu, "PPDP: An efficient and privacy-preserving disease prediction scheme in cloud-based e-healthcare system," *Future Gener. Comput. Syst.*, vol. 79, pp. 16–25, 2018.

[8] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, "HealthDep: An efficient and secure deduplication scheme for cloud-assisted eHealth systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 9, pp. 4101–4112, Sep. 2018.

[9] X. Chen, J. Li, J. Weng, J. Ma, and W. Lou, "Verifiable computation over large database with incremental updates," *IEEE Trans. Comput.*, vol. 65, no. 10, pp. 3184–3195, Oct. 2016.

[10] Y. Rahulamathavan, S. Veluru, J. Han, F. Li, M. Rajarajan, and R. Lu, "User collusion avoidance scheme for privacy-preserving decentralized key-policy attribute-based encryption," *IEEE Trans. Comput.*, vol. 65, no. 9, pp. 2939–2946, Sep. 2016.

[11] O. Ohrimenko *et al.*, "Oblivious multi-party machine learning on trusted processors," in *Proc. 25th USENIX Conf. Secur. Symp.*, 2016, pp. 619–636.

[12] K. A. Jagadeesh, D. J. Wu, J. A. Birgmeier, D. Boneh, and G. Bejerano, "Deriving genomic diagnoses without revealing patient genomes," *Science*, vol. 357, no. 6352, pp. 692–695, 2017.

[13] 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, Sep./Oct. 2014.

[14] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2015, pp. 1–14.

[15] J.-C. Bajard, P. Martins, L. Sousa, and V. Zucca, "Improving the efficiency of SVM classification with FHE," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1709–1722, 2020.

[16] 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.

[17] H. Yunhong, F. Liang, and H. Guoping, "Privacy-preserving SVM classification on vertically partitioned data without secure multiparty computation," in *Proc. IEEE 5th Int. Conf. Natural Comput.*, 2009, pp. 543–546.

[18] H. Zhu, X. Liu, R. Lu, and H. Li, "Efficient and privacy-preserving online medical prediagnosis framework using nonlinear SVM," *IEEE J. Biomed. Health Inform.*, vol. 21, no. 3, pp. 838–850, May 2017.

[19] R. A. Popa, F. H. Li, and N. Zeldovich, "An ideal-security protocol for order-preserving encoding," in *Proc. IEEE Symp. Secur. Privacy*, 2013, pp. 463–477.

[20] F. Kerschbaum, "Frequency-hiding order-preserving encryption," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 656–667.

[21] J. Liang, Z. Qin, S. Xiao, J. Zhang, H. Yin, and K. Li, "Privacy-preserving range query over multi-source electronic health records in public clouds," *J. Parallel Distrib. Comput.*, vol. 135, pp. 127–139, 2020.

[22] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *J. Mach. Learn. Res.*, vol. 12, pp. 1069–1109, 2011.

[23] B. I. Rubinstein, P. L. Bartlett, L. Huang, and N. Taft, "Learning in a large function space: Privacy-preserving mechanisms for SVM learning," *J. Privacy Confidentiality*, vol. 4, no. 1, pp. 65–100, 2012.

[24] J. Zhang, X. Xiao, Y. Yang, Z. Zhang, and M. Winslett, "PrivGene: Differentially private model fitting using genetic algorithms," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2013, pp. 665–676.

[25] R. Chen, Q. Xiao, Y. Zhang, and J. Xu, "Differentially private high-dimensional data publication via sampling-based inference," in *Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2015, pp. 129–138.

[26] N. Cheng *et al.*, "Big data driven vehicular networks," *IEEE Netw.*, vol. 32, no. 6, pp. 160–167, Nov./Dec. 2018.

[27] F. Lyu *et al.*, "Characterizing urban vehicle-to-vehicle communications for reliable safety applications," *IEEE Trans. Intell. Transportation Syst.*, vol. 21, no. 6, pp. 2586–2602, Jun. 2020.

[28] C. Huang, R. Lu, X. Lin, and X. Shen, "Secure automated valet parking: A privacy-preserving reservation scheme for autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11 169–11 180, Nov. 2018.

[29] Y. Zhang, C. Xu, H. Li, K. Yang, N. Cheng, and X. S. Shen, "PROTECT: Efficient password-based threshold single-sign-on authentication for mobile users against perpetual leakage," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2020.2975792.

[30] A. Yang, J. Xu, J. Weng, J. Zhou, and D. S. Wong, "Lightweight and privacy-preserving delegatable proofs of storage with data dynamics in cloud storage," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2018.2851256.

[31] H. Ren, H. Li, D. Liu, G. Xu, N. Cheng, and X. S. Shen, "Privacy-preserving efficient verifiable deep packet inspection for cloud-assisted middlebox," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2020.2991167.

[32] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, 1999, pp. 223–238.

[33] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, 2009, pp. 169–178.

[34] N. G. Tsoutsos and M. Maniatakos, "Efficient detection for malicious and random errors in additive encrypted computation," *IEEE Trans. Comput.*, vol. 67, no. 1, pp. 16–31, Jan. 2018.

[35] X. Li, Y. Zhu, J. Wang, Z. Liu, Y. Liu, and M. Zhang, "On the soundness and security of privacy-preserving SVM for outsourcing data classification," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 5, pp. 906–912, Sep./Oct. 2018.

[36] X. Fu, C. Ong, S. Keerthi, G. G. Hung, and L. Goh, "Extracting the knowledge embedded in support vector machines," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2004, pp. 291–296.

[37] B. Wang, M. Li, and L. Xiong, "FastGeo: Efficient geometric range queries on encrypted spatial data," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 2, pp. 245–258, Mar./Apr. 2019.

[38] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. London, U.K./Boca Raton, FL, USA: Chapman & Hall/CRC, 2007.
[39] B. Wang, M. Li, and H. Wang, "Geometric range search on encrypted spatial data," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 704–719, Apr. 2016.
[40] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Math.*, vol. 1, no. 4, pp. 485–509, 2004.
[41] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 988–999, Sep. 1999.
[42] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *J. Comput. Secur.*, vol. 19, no. 5, pp. 895–934, 2011.
[43] S. Wu, Q. Li, G. Li, D. Yuan, X. Yuan, and C. Wang, "ServeDB: Secure, verifiable, and efficient range queries on outsourced database," in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2019, pp. 626–637.
[44] H. Ren, H. Li, Y. Dai, K. Yang, and X. Lin, "Querying in Internet of Things with privacy preserving: Challenges, solutions and opportunities," *IEEE Netw.*, vol. 32, no. 6, pp. 144–151, Nov./Dec. 2018.

Jinwen Liang received the BS degree in information security from Hunan University, Changsha, China, in 2015. He is currently working toward the PhD degree with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. He is also a visiting PhD student at BBCR Lab, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada. His research interests include applied cryptography, blockchain, order-preserving encryption, and secure data classification.

Zheng Qin (Member, IEEE) received the PhD degree in computer science from Chongqing University, Chongqing, China, in 2001. He was a visiting scholar with Michigan State University from 2010 to 2011. He is a full professor and vice dean with the College of Computer Science and Electronic Engineering, Hunan University, China. He is the director of the Hunan Key Laboratory of Big Data Research and Application, and the vice director of the Hunan Engineering Laboratory of Authentication and Data Security, Changsha, China. His research interests include blockchain, data science, information security, and software engineering.

Jianbing Ni (Member, IEEE) received the BE and ME degrees from the School of Computer Science and Technology, University of Electronic Science and Technology of China, Chengdu, China, in 2011 and 2014, respectively, and the PhD degree from the University of Waterloo, Waterloo, Canada, in 2018. He is currently an assistant professor with the Department of Electrical and Computer Engineering, Queen's University. He was a postdoctoral fellow with the Department of Electrical and Computer Engineering, University of Waterloo from September 2018 to June 2019. His research interests include blockchain, privacy-preserving machine learning, and applied cryptography.

Xiaodong Lin (Fellow, IEEE) received the PhD degree in information engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 1998, and the PhD degree in electrical and computer engineering from the University of Waterloo, Waterloo, Canada, in 2008. He is currently a tenured associate professor with the School of Computer Science, University of Guelph. His research interests include wireless network security, applied cryptography, computer forensics, and software security.

Xuemin Shen (Fellow, IEEE) received the PhD degree in electrical engineering from Rutgers University, New Brunswick, New Jersey, in 1990. He is currently a university professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on network resource management, wireless network security, social networks, 5G and beyond, and vehicular ad hoc and sensor networks. He is a registered professional engineer of Ontario, Canada, an Engineering Institute of Canada fellow, a Canadian Academy of Engineering fellow, a Royal Society of Canada fellow, a Chinese Academy of Engineering foreign fellow, and a distinguished lecturer of the IEEE Vehicular Technology Society and Communications Society. He received the R.A. Fessenden Award, in 2019 from IEEE, Canada, James Evans Avant Garde Award, in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award, in 2015 and Education Award, in 2017 from the IEEE Communications Society. He has also received the Excellent Graduate Supervision Award, in 2006 and Outstanding Performance Award five times from the University of Waterloo and the Premier's Research Excellence Award (PREA), in 2003 from the Province of Ontario, Canada. He has served as the technical program committee chair/co-chair for the IEEE Globecom'16, IEEE Infocom'14, IEEE VTC'10 Fall, IEEE Globecom'07, symposia chair for the IEEE ICC'10, tutorial chair for the IEEE VTC'11 Spring, and chair for IEEE Communications Society Technical Committee on Wireless Communications. He is the editor-in-chief of the *IEEE Internet of Things Journal* and the vice president on Publications of the IEEE Communications Society.