On the Soundness and Security of Privacy-Preserving SVM for Outsourcing Data Classification

Xingxin Li, Youwen Zhu, Jian Wang, Zhe Liu, Yining Liu, Mingwu Zhang

Abstract—Recently, Rahulamathavan *et al.* (IEEE Transactions on Dependable and Secure Computing, 2014, 11(5): 467-479) propose a privacy preserving scheme for outsourcing SVM classification. Their core contribution is a secure protocol to attain the sign of Pailler encrypted numbers. In this paper, we observe that Rahulamathavan *et al.*'s protocol will suffer from some soundness and security problems. Then, we propose a new scheme to securely obtain the encrypted numbers' sign. Theoretical analysis and experiment results show our proposed scheme can not only fix the soundness and security problems, but also achieve higher efficiency.

Index Terms—privacy-preserving, classification, SVM, Paillier encryption

1 Introduction

Support vector machine (SVM) is a widely-used and power tool for data classification, and it has been applied in scientific and engineering problems, such as text classification [5], time series prediction [6], MAC protocol identification [7], fault diagnosis [8]. Generally, SVM classification consists of two stages: training stage and testing stage. The training stage uses labeled samples to gain the classification parameters. Then, testing stage utilizes the classification parameters to predict the class label for any unlabeled sample. Nevertheless, it requires a large number of valid labeled data samples to build a good SVM classifier, which may be of big difficulty for individuals and small organizations. Thus, the individuals and organizations cannot construct their own classifier, and they need other party's SVM classifier to figure out the class labels for their unlabeled data samples. Here, we denote the party holding SVM classifier as a server, and let users denote the individuals/organizations who want to use the server's classifier. The classifier sharing will bring much convenience to users. However, it will present data privacy risk to both the users and the server. The unlabeled samples and their classification results may contain the privacy of users, such as the users' health information, which cannot be disclosed to the server. Besides, the server would be reluctant to directly release his SVM classifier, as the classification parameters of the SVM classifier are his private data and the server has cost much time, money and other resources to gain them.

For dealing with the dilemma, Rahulamathavan *et al.* [1] recently propose a privacy-preserving scheme to support SVM classification in the above scenario while keeping the data of users and server private. Concretely, the unlabeled sample and its classification result is known to its owner (a user), and any classification parameter of the SVM classifier is kept private to the server. In Rahulamathavan *et al.*'s scheme, the unlabeled sample is encrypted by the user using Paillier encryption system [2], then, the user obtains his classification result through a mutual protocol with the server. Rahulamathavan *et al.* declare their core contribution in [1] is a secure protocol to attain the sign of Paillier encrypted numbers.

In this paper, we observe that Rahulamathavan *et al.*'s core protocol for obtaining encrypted numbers' sign is faced with soundness and security problems. The problems will result in that Rahulamathavan *et al.*'s scheme may either return error classification result, or disclose some private information of the server. Then, we propose a new protocol to securely and correctly work out the sign of any Paillier encrypted number. Additionally, through theoretical analysis and simulation experiments, we indicate our proposed scheme can fix the problems in [1] and achieve higher efficiency than Rahulamathavan *et al.*'s scheme.

The rest of the paper is organized as follows. Section 2 reviews the steps of Rahulamathavan *et al.*'s

- *X. Li, Y. Zhu, J. Wang and Z. Liu are with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China.*
Y. Zhu is also with the Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210023, China, and Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China.
Y. Liu is with the School of Computer and Information Security, Guilin University of Electronic Technology, Guilin 541004, China.
M. Zhang is with the School of Computer, Hubei University of Technology, Wuhan 430068, China.
Y. Zhu and Z. Liu are the corresponding authors. E-mail: zhuyw@nuaa.edu.cn, z446liu@uwaterloo.ca

core protocol. Section 3 points out the soundness and security problems of Rahulamathavan *et al.*'s scheme. Section 4 proposes our new secure protocol to efficiently and correctly determine the sign of an encrypted number. Section 5 provides theoretical analysis and experiment results, and compares our proposed prptocol with Rahulamathavan *et al.*'s scheme. At last, Section 6 concludes this paper.

2 Review of Rahulamathavan *et al.*'s Scheme

SVM can be utilized to both two-class problem and multi-class problem. The solution of two-class problem can be employed as the basic building to solve multi-class problem. Here, we only discuss the privacy-preserving SVM for two-class problem. The schemes for multi-class problem can be coped with in a similar way.

Assume the server has the training data set of n d-dimensional labeled samples $\{\widetilde{\boldsymbol{x}}_1, \widetilde{\boldsymbol{x}}_2, \cdots, \widetilde{\boldsymbol{x}}_n\}$, and $y_i \in \{+1, -1\}$ is the class label of sample $\widetilde{\boldsymbol{x}}_i$ for $i = 1$ to n. The server has normalized the training samples and trained a SVM classifier using the normalized data. Let $\overline{\boldsymbol{x}}$ and σ be the mean and standard deviation of the training samples, and $\boldsymbol{x}_i$ denote the normalized result of $\widetilde{\boldsymbol{x}}_i$ for $i = 1$ to n. It has

$$\boldsymbol{x}_i = \frac{\widetilde{\boldsymbol{x}}_i - \overline{\boldsymbol{x}}}{\sigma^2}.$$

The client has an unlabeled sample $\widetilde{\boldsymbol{t}}$ with the same form as the training samples. For classifying the sample, it need to be normalized by the following manner

$$\boldsymbol{t} = \frac{\widetilde{\boldsymbol{t}} - \overline{\boldsymbol{x}}}{\sigma^2}.$$

Then, the class label will be set as the sign of its decision function $d(\boldsymbol{t})$ in SVM. While the training samples are linearly separable, the decision function is

$$d(\boldsymbol{t}) = \sum_{i \in S} \alpha_i y_i (\boldsymbol{x}_i \cdot \boldsymbol{t}) + b. \tag{1}$$

Here, α_is are Lagrangian variable and $\boldsymbol{x}_i$s are support vectors. While the training samples are not linearly separable, some nonlinear kernel function is used to replace the scalar product $(\boldsymbol{x}_i \cdot \boldsymbol{t})$ in (1), i.e.,

$$d(\boldsymbol{t}) = \sum_{i \in S} \alpha_i y_i \kappa(\boldsymbol{x}_i, \boldsymbol{t}) + b, \tag{2}$$

where $\kappa(\cdot, \cdot)$ denotes the kernel function. In [1], Rahulamathavan *et al.* consider a polynomial kernel $\kappa(\boldsymbol{x}_i, \boldsymbol{t}) = (\boldsymbol{x}_i \cdot \boldsymbol{t} + 1)^p$.

To protect the private information of $\widetilde{\boldsymbol{t}}$, the client will encrypt each dimension of $\widetilde{\boldsymbol{t}}$ using Paillier encryption system [2], and then submit the encrypted sample to the server. Let $[\![m]\!]$ denote the ciphertext of any plaintext m in Paillier encryption system, and the encrypted sample $[\![\widetilde{\boldsymbol{t}}]\!]$ equals to the encrypted vector $([\![\widetilde{t}_1]\!], [\![\widetilde{t}_2]\!], \cdots, [\![\widetilde{t}_d]\!])$ where $\widetilde{t}_i$ denotes the i-th dimension of the sample $\widetilde{\boldsymbol{t}}$ for $i = 1$ to d. Paillier encryption system is additively homomorphic, and it satisfies the following properties.

$$[\![m_1 + m_2 \pmod N]\!] = [\![m_1]\!] * [\![m_2]\!], \tag{3}$$

$$[\![\alpha * m_1 \pmod N]\!] = [\![m_1]\!]^{\alpha}, \tag{4}$$

where N equals to the product of two large prime numbers p and q, and $\mathbb{Z}_N$ is plaintext space of Paillier encryption system. Besides, m_1, m_2 are any numbers belonging to $\mathbb{Z}_N$, and α is an integer. Since the plaintext space of Paillier encryption system is a subset of integers, Rahulamathavan *et al.* [1] use a scaling factor γ to scale each dimension of the samples. To reduce its negative influence on classification accuracy, the scaling factor γ is generally set to 10^6 or larger.

After receiving the encrypted sample $[\![\widetilde{\boldsymbol{t}}]\!]$, the server in [1] will normalize and scale it in encrypted domain. Then, based on the additively homomorphic property shown in equations (3) and (4), taking $[\![\widetilde{\boldsymbol{t}}]\!]$ and the classification parameters as input, the server can locally gain $[\![d(\boldsymbol{t}) \pmod N]\!]$. For more detail about the steps of computing $[\![d(\boldsymbol{t}) \pmod N]\!]$, please refer to [1]. Here, we focus on the core phase of Rahulamathavan *et al.*'s scheme, by which, the client will obtain the sign of decision function $d(\boldsymbol{t})$. Besides, the client can learn nothing about $d(\boldsymbol{t})$ apart from its sign, and the server can attain nothing about the client's private input and output.

Assume $|d(\boldsymbol{t})| < 10^l$, in which l is a positive integer. In fact, due to the scaling factor γ, the original decision function will be multiplied by a large number. For example, while using the polynomial kernel $\kappa(\boldsymbol{x}_i', \boldsymbol{t}) = (\boldsymbol{x}_i' \boldsymbol{t} + 1)^p$, the decision function $d(\boldsymbol{t})$ is multiplied by γ^{2p+1} as shown in [1]. Thus, 10^l is a large number during computing the decision function's sign in fourth step. Concretely, Rahulamathavan *et al.*'s approach to obtain the sign proceeds as follows.

Firstly, the server computes a new value

$$[\![z]\!] = [\![10^l]\!] * [\![d(\boldsymbol{t}) \pmod N]\!]. \tag{5}$$

That is, $[\![z]\!] = [\![10^l + d(\boldsymbol{t})]\!]$ and $z = 10^l + d(\boldsymbol{t})$. Since $|d(\boldsymbol{t})| < 10^l$, then $0 < z < 2 * 10^l$. Hence, the most significant (decimal) digit of z equals to

$$\widetilde{z} = 10^{-l} * \big(z - (z \bmod 10^l)\big). \tag{6}$$

Besides, if $d(\boldsymbol{t}) < 0$, it has $z \bmod 10^l = 10^l + d(\boldsymbol{t}) = z$; otherwise, $z \bmod 10^l = d(\boldsymbol{t}) = z - 10^l$. Thus,

$$\begin{cases} \widetilde{z} = 0, & \text{if } d(\boldsymbol{t}) < 0, \\ \widetilde{z} = 1, & \text{otherwise.} \end{cases} \tag{7}$$

If $class_+$ and $class_-$ denotes the class label of $sign(d(\boldsymbol{t})) = +1$ and $sign(d(\boldsymbol{t})) = -1$, respectively. Then, the test sample's class label will be

$$M_c = \widetilde{z} * (class_+ - class_-) + class_-. \tag{8}$$

Secondly, the server perturbs z with a random number r as follows.

$$[\![z_r]\!] = [\![z]\!] * [\![r]\!]. \tag{9}$$

Here, z_r is expected to be $(z + r)$ in [1].

Thirdly, the client decrypts $[\![z_r]\!]$, computes the value $(z_r \bmod 10^l)$, and returns $[\![z_r \bmod 10^l]\!]$ to the server.

Fourthly, the server taking $(r \bmod 10^l)$ as input and the client taking $(z_r \bmod 10^l)$ as input collaboratively implement secure comparison protocol [3], [4] in plaintext domain, such that the server obtain the encrypted comparison result $[\![\lambda]\!]$, where $\lambda = 0$, if $(z_r \bmod 10^l) \geqslant (r \bmod 10^l)$; $\lambda = 1$, otherwise. As Rahulamathavan *et al.* regard z_r as $(z + r)$, in this sub-step, they indeed expect to obtain $\lambda = 0$, if $\big((z + r) \bmod 10^l\big) \geqslant (r \bmod 10^l)$; $\lambda = 1$, otherwise.

Lastly, the server successively computes

$$\mathcal{X} = [\![z_r \bmod 10^l]\!] * [\![r \bmod 10^l]\!]^{-1} * [\![\lambda]\!]^{10^l}, \tag{10}$$

$$[\![\widetilde{z}]\!] = \big([\![z]\!] * \mathcal{X}^{-1}\big)^{10^{-l}}, \tag{11}$$

$$[\![M_c]\!] = [\![\widetilde{z}]\!]^{(class_+ - class_-)} * [\![class_-]\!]. \tag{12}$$

After that, the server returns $[\![M_c]\!]$ to the client, and the latter decrypts it and obtain the private classification result M_c which completes the entire protocol. It should be noted that Rahulamathavan *et al.* expect to compute $[\![z \bmod 10^l]\!]$ in equation (10). Here, we use the identifier $\mathcal{X}$ at the left side of equation (10) instead of $[\![z \bmod 10^l]\!]$ used in [1].

The correctness of Rahulamathavan *et al.*'s scheme depends on that the plaintext hidden in $\mathcal{X}$ in equation (10) just right equals to $(z \bmod 10^l)$. Namely, it requires

$$\mathcal{X} = [\![z \bmod 10^l]\!]. \tag{13}$$

If equation (13) is correct, based on equation (6) and additively homomorphic property, it can ensure $\big([\![z]\!] * \mathcal{X}^{-1}\big)^{10^l} = \big([\![z]\!] * [\![z \bmod 10^l]\!]^{-1}\big)^{10^l} = [\![\widetilde{z}]\!]$, i.e., equation (11) holds. Further, equation (12) can compute the ciphertext of correct class label, on the basis of equation (8).

However, in this paper, we observe that equation (13) does not always hold, and it will bring about some soundness and security problems to Rahulamathavan *et al.*'s protocol. The detail will be presented in Section 3.

3 Soundness and Security Problems of Rahulamathavan *et al.*'s Scheme

In this section, we will reconsider the soundness and security of of Rahulamathavan *et al.*'s scheme in [1].

3.1 Soundness

It is easy to say that equations (6), (7), (8) are correct. Equation (9) is used to perturb z. In the following, we will analyze equation (10).

For any integer z, r, it has

$$z \bmod 10^l = \Big(\big((z+r) \bmod 10^l\big) - (r \bmod 10^l)\Big) \bmod 10^l.$$

If setting

$$\lambda = \begin{cases} 0, & \text{if } \big((z + r) \bmod 10^l\big) \geqslant (r \bmod 10^l) \\ 1, & \text{if } \big((z + r) \bmod 10^l\big) < (r \bmod 10^l) \end{cases}, \tag{14}$$

then, it will be $(z \bmod 10^l) = \big((z + r) \bmod 10^l\big) - (r \bmod 10^l) + \lambda * 10^l$. Based on additively homomorphic property, $[\![(z+r) \bmod 10^l]\!] * [\![r \bmod 10^l]\!]^{-1} * [\![\lambda]\!]^{10^l}$ will be the corresponding ciphertext of $(z \bmod 10^l)$.

However, Rahulamathavan *et al.* [1] simply use $[\![z_r \bmod 10^l]\!]$ instead of $[\![(z + r) \bmod 10^l]\!]$ during computing $\mathcal{X}$ in equation (10). It cannot ensure $\mathcal{X}$ to be $[\![z \bmod 10^l]\!]$, since $(z_r \bmod 10^l) = \big((z+r) \bmod 10^l\big)$ does not always hold.

According to equation (9), it has $z_r = (z+r) \bmod N$ where $\mathbb{Z}_N$ is the plaintext space of Paillier encryption system. Due to the overflow, it will be $z_r = z + r - N$, if $z + r \geqslant N$; and $z_r = z + r$, otherwise. Because $N \bmod 10^l \neq 0$, it must be $(z_r \bmod 10^l) \neq \big((z + r) \bmod 10^l\big)$ while $z + r \geqslant N$. Besides, λ will not equal to the comparison result of $\big((z + r) \bmod 10^l\big)$ and $(r \bmod 10^l)$ with high probability in the overflow situation. That is, the value of λ will be different from that defined in equation (14). Further, it will result in wrong $\big((z + r) \bmod 10^l\big)$, $\widetilde{z}$, M_c in equations (10), (11), (12). As a consequence, the client cannot obtain a correct classification result. In the following, we give a toy example to demonstrate the classification error on account of overflow.

Toy example 1. Assume $d(\boldsymbol{t}) = 8$, $N = 3 * 11 = 33$, $10^l = 10$ (i.e., $l = 1$). Then, $z = d(\boldsymbol{t}) + 10^l = 18$. If the client selects $r = 23$, it would be $[\![z + r]\!] = [\![8]\!]$, i.e., $z_r = 8$. It is easy to say that $(z + r) \bmod 10^l = 1$, but $z_r \bmod 10^l = 8$.

Since $(z_r \bmod 10^l) > (r \bmod 10^l)$, it will lead to $\lambda = 0$, and

$$\mathcal{X} = [\![8]\!] * [\![3]\!]^{-1} * [\![0]\!]^{10} = [\![5]\!],$$

$$[\![\widetilde{z}]\!] = \big([\![z]\!] * \mathcal{X}^{-1}\big)^{10^{-l}} = [\![31]\!],$$

$$[\![M_c]\!] = [\![\widetilde{z}]\!]^{(class_+ - class_-)} * [\![class_-]\!] = [\![31 * class_+ - 30 * class_-]\!].$$

However, the correct values should be $\lambda = 1$, and

$$\mathcal{X} = [\![1]\!] * [\![3]\!]^{-1} * [\![1]\!]^{10} = [\![8]\!],$$

$$[\![\widetilde{z}]\!] = \big([\![z]\!] * \mathcal{X}^{-1}\big)^{10^{-l}} = [\![1]\!],$$

$$[\![M_c]\!] = [\![\widetilde{z}]\!]^{(class_+ - class_-)} * [\![class_-]\!] = [\![class_+]\!].$$

Namely, Rahulamathavan *et al.*'s scheme returns an incorrect classification result to the client, while overflow occurs in the toy example.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2682244, IEEE Transactions on Dependable and Secure Computing

Therefore, the scheme in [1] has the soundness problems due to possible overflow.

3.2 Security of a Plausible Solution

From equation (5) and $|d(\boldsymbol{t})| < 10^l$, it can be deduced that $z = 10^l + d(\boldsymbol{t})$, $0 < z < 2 * 10^l$. If the server only selects $0 \leqslant r \leqslant N - 2 * 10^l$, it will be $0 < z + r < N$ which can avoid the overflow error. Nevertheless, due to the contraction of r's value domain, the client may infer some information about $d(\boldsymbol{t})$ with high probability, i.e., it will violate the server's privacy. The reasons are as follows.

In [1], the server reveals $z_r = z + r$ to the client. As $d(\boldsymbol{t}) = z - 10^l$ and l is a public number, we just analyze the possible disclosure of z due to z_r. Without loss of generality, we assume that z is with uniform distribution, i.e., $P(z{=}i) = \frac{1}{2*10^l-1}$ for each $i = 1, 2, \cdots, 2{*}10^l{-}1$. Then, we say z_r can securely protect z, only if $|P(z{=}i|z_r) - P(z{=}i)|$ (i.e., $|P(z{=}i|z_r){-}\frac{1}{2*10^l-1}|$) is negligibly small for any $i = 1, 2, \cdots, 2{*}10^l{-}1$ and $z_r = 1, 2, \cdots, N{-}1$. Here, $P(z{=}i|z_r)$ represents the conditional probability of $z = i$ given z_r. We will show the scheme in [1] cannot satisfy the above condition.

After gaining z_r, based on $0 \leqslant r \leqslant N{-}2{*}10^l$ and $z = z_r - r$, the client can deduce

$$z_r + 2 * 10^l - N \leqslant z \leqslant z_r.$$

Besides, $0 < z < 2 * 10^l$. Let $B_L = \max\{z_r{+}2{*}10^l{-}N, 1\}$ and $B_R = \min\{2 * 10^l - 1,\ z_r\}$. Then,

$$B_L \leqslant z \leqslant B_R.$$

Let the set $\boldsymbol{D} = \{x \in \mathbb{Z} | 0 {<} x {<} B_L$, or $B_R {<} x {<} 2{*}10^l\}$. The client can conclude, for any $i \in \boldsymbol{D}$,

$$P(z = i|z_r) = 0.$$

While $z_r > N + 1 - 2 * 10^l$ or $z_r < 2 * 10^l - 1$, it has $B_L > 1$ or $B_R < 2{*}10^l - 1$, which will result in $\boldsymbol{D} \neq \emptyset$. Therefore, if $z_r > N{+}1{-}2{*}10^l$ or $z_r < 2{*}10^l{-}1$, it must exist some $i \in \boldsymbol{D}$ such that $|P(z{=}i|z_r) - P(z{=}i)| = |0{-}\frac{1}{2*10^l-1}| = \frac{1}{2*10^l-1}$ is not negligible, i.e., z_r cannot well hide z in this situation.

Intuitively, while $\boldsymbol{D}{\neq}\emptyset$, it will lessen the value range of z from $\{x \in \mathbb{Z} | 0 {<} x {<} 2{*}10^l\}$ to $\{x \in \mathbb{Z} | B_L {\leqslant} x {\leqslant} B_R\}$. Accordingly, the number of z's possible values will be $(B_R - B_L + 1)$, once z_r is revealed to the client. Fig. 1 vividly shows the number of z's possible values when z_r varies from 1 to $N{-}1$. As can be seen, the client can learn some extra information about z in $\frac{4*10^l}{N}{*}100\%$ percent of z_r's value range, if setting $0 \leqslant r \leqslant N{-}2{*}10^l$. Note that the server has r, but does not know the value of z. Thus, the server has no idea about which range the value of z_r really belongs to in each implementation.

For a polynomial kernel function $\kappa(\boldsymbol{x}_i, \boldsymbol{t}) = (\boldsymbol{x}_i \cdot \boldsymbol{t}{+}1)^p$, its value exponentially increases along with the degree p. We have simulated the value over a real dataset "a1a" from LIBSVM data set [12]. The results

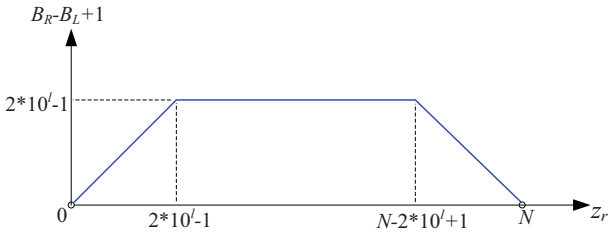


Fig. 1. The number of z's possible values after z_r (i.e., $z + r$) is disclosed, if it is set $0 \leqslant r \leqslant N - 2 * 10^l$. Here, $(B_R - B_L + 1)$ equals to the number of z's possible values.

show that the upper bound of z is about 2^{1016} while $p = 5$ and the scale factor is set to 10^5. Accordingly, $\frac{4*2^{1016}}{N}{*}100\%$ will be 2% around if the bit length of key is 1024. To improve robustness, the scheme should tolerate a wide range of the degree p. While p or the scale factor is larger in some applications, the percent will increase fast. Even if the variable z is not big, such as $1 \leqslant z \leqslant 10^{100}$, and the range of random variable r is set to $0 \leqslant r \leqslant 10^{100}$, the value $(z + r)$ will still disclose some information about the private data z, since the client can utilize $(z{+}r)$ to narrow down the space of possible values of z. Here, we provide a toy example to indicate the privacy leakage.

Toy example 2. Assume $1 \leqslant z \leqslant 5$, $N = 5 * 7 = 35$, $0 \leqslant r \leqslant 5$, then, it will be $1 \leqslant z{+}r \leqslant 10$. If the value of $(z{+}r)$ is leaked, it will result in the disclosure of z with a high probability. As shown in Table 1, if $z{+}r < 5$ or $z{+}r > 6$, the value range of unknown z will be smaller than the initial $\{1, 2, 3, 4, 5\}$. For instance, if $z + r = 9$ and it is leaked to the client, then the client can infer $z = 9 - r$. Besides, $-5 \leqslant -r \leqslant 0$. Thus, $z \geqslant 9 - 5 = 4$, i.e., z must be 4 or 5.

Totally, there are 10 cases for the value of $(z{+}r)$ in the toy example 2, but $(z + r)$ in 8 cases will bring about information disclosure of private number z, since the space of possible values of z can be narrowed down.

In general, Rahulamathavan *et al.*'s scheme [1] either has soundness problems, or reveals some private information of the server.

TABLE 1
Disclosed value range of z while $(z + r)$ is leaked

$z + r$	Possible values of z, if $(z + r)$ is leaked
1	1
2	1, 2
3	1, 2, 3
4	1, 2, 3, 4
5	1, 2, 3, 4, 5
6	1, 2, 3, 4, 5
7	2, 3, 4, 5
8	3, 4, 5
9	4, 5
10	5

4 Our Proposed Scheme

In this section, we will propose a new protocol to securely determine the sign of $d(\boldsymbol{t})$ and correctly attain the classification result M_c. Additionally, our proposed scheme is a hybrid approach utilizing both homomorphic encryption system [2] and garbled circuit protocol [9], [10], and thus costs less computation time than Rahulamathavan *et al.*'s method in [1]. The proposed scheme can also be severed as an efficient basic building in any other application that needs to figure out the sign of a number in encrypted domain.

Here, we assume $2 * 10^l < N$, where $\mathbb{Z}_N$ is the plaintext space of Paillier encryption system. Then, $|d(\boldsymbol{t})| < 10^l \leqslant \frac{N-1}{2}$. In fact, while $|d(\boldsymbol{t})| < 10^l$, the size of value domain of $d(\boldsymbol{t})$ is $2 * 10^l$. It must be set $2 * 10^l < N$, otherwise, the plaintext space cannot cover all possible values of $d(\boldsymbol{t})$. We introduce the main method as follows.

If setting $\mathcal{U} = d(\boldsymbol{t}) + \frac{N-1}{2}$, then it will be $0 < \mathcal{U} < N$. Further, we have

- While $0 < d(\boldsymbol{t})$, there is $\frac{N-1}{2} < \mathcal{U} < N$, i.e., $\frac{N+1}{2} \leqslant \mathcal{U} < N$. It will be $N+1 \leqslant 2{*}\mathcal{U} < 2{*}N$ and $2{*}\mathcal{U} \pmod N = 2{*}\mathcal{U} - N$. The least significant binary digit of $\big(2{*}\mathcal{U} \pmod N\big)$ will be 1.
- While $0 \geqslant d(\boldsymbol{t})$, there is $0 < \mathcal{U} \leqslant \frac{N-1}{2}$. Then, $0 < 2{*}\mathcal{U} < N$ and $2{*}\mathcal{U} \pmod N = 2{*}\mathcal{U}$. The least significant binary digit of $\big(2{*}\mathcal{U} \pmod N\big)$ will be 0.

Therefore, the least important binary digit of $\big(2{*}\mathcal{U} \pmod N\big)$ equals to 1, if and only if $0 < d(\boldsymbol{t})$. Based on the conclusion, we achieve our new scheme Secure Sign Protocol (SSP), which can securely and correctly work out the sign of $d(\boldsymbol{t})$. The detailed steps are shown in Protocol 1.

In our proposed SSP, the server first computes $[\![\mathcal{U}]\!] = [\![d(\boldsymbol{t})]\!] * [\![\frac{N-1}{2}]\!]$. Based on the homomorphic property, we have $\mathcal{U} = d(\boldsymbol{t}) + \frac{N-1}{2} \pmod N$. It is easy to say $0 < d(\boldsymbol{t}) + \frac{N-1}{2} < N$, thus $\mathcal{U}$ exactly equals to $(d(\boldsymbol{t}) + \frac{N-1}{2})$. By setting $[\![\mathcal{T}]\!] = [\![\mathcal{U}]\!]^2$, it will be $\mathcal{T} = 2{*}\mathcal{U} \pmod N$. Besides, it must be $\mathcal{T} \neq 0$, since $2{*}\mathcal{U} \neq N$ always holds. Thus, $0 < \mathcal{T} < N$. As we have aforementioned, the least significant binary digit of $\mathcal{T}$ equals to 1, if and only if $0 < d(\boldsymbol{t})$.

Next, the lines 2 to 6 of SSP aim at computing the least significant bit of $\mathcal{T}$ in encrypted domain. To protect $\mathcal{T}$, the server returns $[\![\mathcal{V}]\!] = [\![\mathcal{T}]\!] * [\![\mathcal{R}]\!]$ to the client, where $\mathcal{R}$ is randomly selected from $\mathbb{Z}_N$. After decrypting $[\![\mathcal{V}]\!]$, the client can obtain $\mathcal{V} = (\mathcal{T} + \mathcal{R}) \bmod N$, where $0 < \mathcal{T} + \mathcal{R} < 2{*}N$. Then,

- If $N \leqslant \mathcal{T} + \mathcal{R} < 2{*}N$, it has $\mathcal{V} = \mathcal{T} + \mathcal{R} - N$. As $0 < \mathcal{T} < N$, thus $\mathcal{V} < \mathcal{R}$.
- If $0 < \mathcal{T} + \mathcal{R} < N$, it has $\mathcal{V} = \mathcal{T} + \mathcal{R}$. Thus $\mathcal{V} > \mathcal{R}$, since $0 < \mathcal{T}$.

Therefore, $\mathcal{V} = \mathcal{T} + \mathcal{R} - N$ if and only if $\mathcal{V} < \mathcal{R}$, otherwise $\mathcal{V} = \mathcal{T} + \mathcal{R}$. Let $\mathcal{V}_0$, $\mathcal{T}_0$ and $\mathcal{R}_0$ denote the least significant bits of $\mathcal{V}$, $\mathcal{T}$ and $\mathcal{R}$. We further discuss the significant bits in the following.

Protocol 1 Secure Sign Protocol (SSP)

Input: The client (**Cl**) has the secret key sk of Paillier encryption system, and pk is the corresponding public key. The server (**Se**) has class labels $class_+$, $class_-$, the ciphertext $[\![d(\boldsymbol{t})]\!]$ encrypted by using pk. Besides, $|d(\boldsymbol{t})| < 10^l$, and $2 * 10^l < N$.

Output: **Cl** obtains the the matching class label M_c, where $M_c = class_+$ if $d(\boldsymbol{t}) > 0$, otherwise $M_c = class_-$. Note that **Cl** should only get M_c while both parties learn nothing about $d(\boldsymbol{t})$.

1: **Se** computes $[\![\mathcal{U}]\!] = [\![d(\boldsymbol{t})]\!] * [\![\frac{N-1}{2}]\!]$, and sets $[\![\mathcal{T}]\!] = [\![\mathcal{U}]\!]^2$.

2: **Se** selects a random number $\mathcal{R} \in \mathbb{Z}_N$, computes $[\![\mathcal{V}]\!] = [\![\mathcal{T}]\!] * [\![\mathcal{R}]\!]$, and sends $[\![\mathcal{V}]\!]$ to **Cl**.

3: **Cl** decrypts $[\![\mathcal{V}]\!]$, obtains $\mathcal{V} = (\mathcal{T}+\mathcal{R}) \bmod N$, and computes $\mathcal{V}_0 = \mathcal{V} \bmod 2$, i.e., the least significant bit of $\mathcal{V}$.

4: **Se** selects a random bit $c \in \{0, 1\}$, and constructs a garbled comparison circuit based on the following steps

- Comparing $\mathcal{V}$ and $\mathcal{R}$. Let $\beta = (\mathcal{V} < \mathcal{R})$ denote the comparison result. Namely, $\beta = 1$ if $\mathcal{V} < \mathcal{R}$, otherwise $\beta = 0$. Note that the comparison result β, as part of the circuit, is not known to Client and Server.
- Computing $c \oplus \beta$ using a free XOR gate.

5: **Cl** attains the result $c \oplus \beta$ by evaluating the garbled circuit, and then sends $\mathcal{W} = [\![c \oplus \beta \oplus \mathcal{V}_0]\!]$ to **Se**.

6: **Se** computes $\lambda = c \oplus \mathcal{R}_0$ where $\mathcal{R}_0 = \mathcal{R} \bmod 2$ is the least significant bit of $\mathcal{R}$, and sets

$$\mathcal{Y} = \begin{cases} \mathcal{W}, & \text{if } \lambda = 0, \\ [\![1]\!] * \mathcal{W}^{-1}, & \text{if } \lambda = 1. \end{cases}$$

7: **Se** computes

$$\mathcal{M} = [\![class_-]\!] * \mathcal{Y}^{class_+ - class_-},$$

and sends $\mathcal{M}$ to **Cl**.

8: **Cl** decrypts $\mathcal{M}$, and sets the decryption values as the classification result M_c.

- While $\mathcal{V} = \mathcal{T} + \mathcal{R} - N$. Then, $\mathcal{V}_0 = \mathcal{V} \bmod 2 = (\mathcal{T} + \mathcal{R} - N) \bmod 2 = \big((\mathcal{T} \bmod 2) + (\mathcal{R} \bmod 2) + (N \bmod 2)\big) \bmod 2$. That is, $\mathcal{V}_0 = (\mathcal{T}_0 + \mathcal{R}_0 + 1) \bmod 2$. According to Table 2, we have $\mathcal{V}_0 = 1 \oplus \mathcal{R}_0 \oplus \mathcal{T}_0$. Thus, $\mathcal{T}_0 = 1 \oplus \mathcal{R}_0 \oplus \mathcal{V}_0$.
- While $\mathcal{V} = \mathcal{T} + \mathcal{R}$. Then, $\mathcal{V}_0 = \mathcal{V} \bmod 2 = (\mathcal{T} + \mathcal{R}) \bmod 2 = \big((\mathcal{T} \bmod 2) + (\mathcal{R} \bmod 2)\big) \bmod 2$. That is, $\mathcal{V}_0 = (\mathcal{T}_0 + \mathcal{R}_0) \bmod 2$. Accordingly, we have $\mathcal{V}_0 = \mathcal{R}_0 \oplus \mathcal{T}_0$, i.e., $\mathcal{V}_0 = 0 \oplus \mathcal{R}_0 \oplus \mathcal{T}_0$. Thus, $\mathcal{T}_0 = 0 \oplus \mathcal{R}_0 \oplus \mathcal{V}_0$.

In general, if setting β to be the comparison result of $\mathcal{V}$ and $\mathcal{R}$, s.t.,

$$\beta = \begin{cases} 1, & \text{if } \mathcal{V} < \mathcal{R} \\ 0, & \text{otherwise} \end{cases}, \tag{15}$$

TABLE 2
Truth table for $\big((\mathcal{T}_0+\mathcal{R}_0+1) \bmod 2\big)$ and $(1\oplus\mathcal{R}_0\oplus\mathcal{T}_0)$

$\mathcal{T}_0$	$\mathcal{R}_0$	$(\mathcal{T}_0+\mathcal{R}_0+1) \bmod 2$	$1\oplus\mathcal{R}_0\oplus\mathcal{T}_0$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	1	1

we will have

$$\mathcal{T}_0 = \beta \oplus \mathcal{R}_0 \oplus \mathcal{V}_0. \tag{16}$$

In lines 4 to 5 of our scheme, the server and client will perform a garbled circuit protocol to securely comparison $\mathcal{V}$ and $\mathcal{R}$ in plaintext domain, such that β equals to the comparison result shown in equation (15). In the garbled circuit protocol, the server is the circuit constructor, and the client is the evaluator. To hide the comparison result which may reveal partial information about $\mathcal{T}$, the server flips a coin $c \in \{0,1\}$ to protect β. Then, the client can only attain $c \oplus \beta$ at the end of the garbled circuit protocol. After that, the client can compute $c \oplus \beta \oplus \mathcal{V}_0$. At the other side, the server has $c \oplus \mathcal{R}_0$. Based on equation (16), it is easy to say $\mathcal{T}_0 = (c \oplus \beta \oplus \mathcal{V}_0) \oplus (c \oplus \mathcal{R}_0)$. Besides, for any bit $x \in \{0,1\}$, it is $x \oplus 0 = x$ and $x \oplus 1 = 1 - x$. After the client sends the ciphertext $[\![c \oplus \beta \oplus \mathcal{V}_0]\!]$ to the server, the latter can attain $[\![\mathcal{T}_0]\!]$ to be $[\![c \oplus \beta \oplus \mathcal{V}_0]\!]$ if $c \oplus \mathcal{R}_0 = 0$, or $[\![1]\!] * [\![c \oplus \beta \oplus \mathcal{V}_0]\!]^{-1}$ if $c \oplus \mathcal{R}_0 = 1$. Therefore, $\mathcal{Y}$ is a ciphertext of the correct $\mathcal{T}_0$ in the line 6 of our SSP.

Finally, based on the homomorphic property, we can guarantee $\mathcal{M} = [\![class_+]\!]$ if $\mathcal{T}_0 = 1$, $\mathcal{M} = [\![class_-]\!]$ if $\mathcal{T}_0 = 0$ in the line 7 of SSP. Consequently, the client will receive a correct classification result in our new protocol.

Remark 1. It is impossible $\mathcal{V} = \mathcal{R}$, since $\mathcal{T} \neq 0$. While $\mathcal{V} = \mathcal{R}$, the value of $\beta \oplus \mathcal{R}_0 \oplus \mathcal{V}_0$ may be unequal to the least significant bit of $\mathcal{T}$. Fortunately, our scheme can ensure $\mathcal{T} \neq 0$, because $\mathcal{T} = 2{*}\mathcal{U} \pmod N$ and $2{*}\mathcal{U} \neq N$. Therefore, our proposed SSP will always return an exactly correct result.

Remark 2. The lines 1 to 2 of our SSP can be replaced by a more efficient manner as follows.

- **Se** selects a random number $\mathcal{R} \in \mathbb{Z}_N$, computes $[\![\mathcal{V}]\!] = [\![d(\boldsymbol{t})]\!]^2 * [\![\big((\mathcal{R}-1) \mod N\big)]\!]$, and sends $[\![\mathcal{V}]\!]$ to **C1**.

The lines 6 to 7 of our SSP can be replaced by the following more efficient manner as well.

- **Se** computes

$$\mathcal{M} = \begin{cases} [\![class_-]\!] * \mathcal{W}^{class_+ - class_-}, & \text{if } c \oplus \mathcal{R}_0 = 0 \\ [\![class_+]\!] * \mathcal{W}^{class_- - class_+}, & \text{if } c \oplus \mathcal{R}_0 = 1 \end{cases},$$

  and returns $\mathcal{M}$ to **C1**.

To clearly illustrate our approach, we respectively split the computation of $[\![\mathcal{V}]\!]$ and $\mathcal{M}$ into two lines in Protocol 1.

5 Analysis and Evaluation

5.1 Correctness Analysis

We guarantee the correctness of our proposed scheme by the following theorem.

Theorem 1: (correctness) Assume each participant exactly follow the prescriptive steps in our proposed scheme, i.e., each participant is assumed to be semi-honest, then the client can correctly attain the matching class label.

Proof: In our scheme, it has $\mathcal{U} = d(\boldsymbol{t}) + \frac{N-1}{2}$. While $|d(\boldsymbol{t})| < 10^l \leqslant \frac{N-1}{2}$, we have $0 < \mathcal{U} < N$. Meanwhile, if $d(\boldsymbol{t}) > 0$, it is $N < 2 * \mathcal{U} < 2N$; otherwise, it is $0 < 2 * \mathcal{U} < N$. Let $\mathcal{T} = 2{*}\mathcal{U} \pmod N$. Then $\mathcal{T} = 2 * \mathcal{U} - N$ if $d(\boldsymbol{t}) > 0$, otherwise $\mathcal{T} = 2 * \mathcal{U}$. That is, $\mathcal{T}$ is odd if and only if if $d(\boldsymbol{t}) > 0$. Thus, the least significant bit of $\mathcal{T}$, denoted as $\mathcal{T}_0$, can be used to represent the sign of $d(\boldsymbol{t})$. In our scheme, a random number $\mathcal{R} \in \mathbb{Z}_N$ is utilized to protect $\mathcal{T}$, and the client can obtain just $\mathcal{V} = (\mathcal{R}+\mathcal{U}) \bmod N$. The line 4 of our scheme securely compares $\mathcal{R}$ (held by the server) and $\mathcal{V}$ (held by the client) such that the client attains $c \oplus \beta$ where c is a random bit selected by the server and $\beta = (\mathcal{V} < \mathcal{R})$ is the comparison result. In Table 1 and equation (16), we have shown $\mathcal{T}_0 = \beta \oplus \mathcal{R}_0 \oplus \mathcal{V}_0$ in which $\mathcal{R}_0$ is the least significant bit of $\mathcal{R}$. Thus, $\mathcal{T}_0 = (c \oplus \mathcal{R}_0) \oplus (c \oplus \beta \oplus \mathcal{V}_0)$. Then, $\mathcal{T}_0 = c \oplus \beta \oplus \mathcal{V}_0$ if $c \oplus \mathcal{R}_0 = 0$; $\mathcal{T}_0 = 1 - c \oplus \beta \oplus \mathcal{V}_0$ if $c \oplus \mathcal{R}_0 = 1$. Since $\mathcal{W} = [\![c \oplus \beta \oplus \mathcal{V}_0]\!]$, based on additively homomorphic property, the line 6 of our scheme can correctly gain $\mathcal{Y} = [\![\beta \oplus \mathcal{R}_0 \oplus \mathcal{V}_0]\!]$ in encrypted form, i.e., $\mathcal{Y} = [\![\mathcal{T}_0]\!]$. Furthermore, in line 7, we have

$$\begin{aligned} \mathcal{M} &= [\![class_-]\!] * \mathcal{Y}^{class_+ - class_-} \\ &= [\![class_- + \mathcal{T}_0 * (class_+ - class_-)]\!] \\ &= [\![\mathcal{T}_0 * class_+ + (1 - \mathcal{T}_0) * class_-]\!]. \end{aligned}$$

Therefore, $\mathcal{M} = [\![class_+]\!]$ if $d(\boldsymbol{t}) > 0$ (i.e., $\mathcal{T}_0 = 1$); $\mathcal{M} = [\![class_-]\!]$ if $d(\boldsymbol{t}) < 0$ (i.e., $\mathcal{T}_0 = 0$). Consequently, the client can receive a correct classification result by decrypting $\mathcal{M}$, which completes the proof of Theorem 1. □

5.2 Security Analysis

We consider the security of our scheme under semi-honest model [10]. A formal definition of security against semi-honest adversaries can be described as follows.

Definition 1. *(privacy under semi-honest model [10])* Let $f(x,y)$ be a functionality, and $f_1(x,y)$ (resp. $f_2(x,y)$) denote the first (resp. second) element of $f(x,y)$. Let Π be a two-party protocol for computing $f(x,y)$ such that the first (resp. second) party obtains $f_1(x,y)$ (resp. $f_2(x,y)$). The view of the first (resp. second) party during an execution of Π on (x,y), denoted as $VIEW_1(x,y)$ (resp. $VIEW_2(x,y)$), is $(x, r, m_1, \cdots, m_t)$ (resp. $(y, r, m_1, \cdots, m_t)$), where x (resp. y) represents the input of the first (resp.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2682244, IEEE Transactions on Dependable and Secure Computing

second) party, r represents its random number and m_i represents the i-th message it has received. We say that protocol Π privately computes function f, i.e., Π is secure against semi-honest adversaries, if there exists probabilistic polynomial-time algorithms S_1 and S_2, such that

$$(S_1(x, f_1(x, y)) \stackrel{c}{\equiv} (VIEW_1(x, y)) \quad (17)$$

$$(S_2(y, f_2(x, y)) \stackrel{c}{\equiv} (VIEW_2(x, y)) \quad (18)$$

where $\stackrel{c}{\equiv}$ represents computational indistinguishability.

Based on the above security definition, we prove the security of our SSP through the following theorem.

Theorem 2: Assume each participant is semi-honest, our proposed scheme SSP discloses nothing useful about the privacy of any participant, and is provably secure.

Proof: In our SSP, let the client's (resp. the server's) view be denoted by $VIEW_1$ (resp. $VIEW_2$). From SSP, we can obtain $VIEW_1 = (\mathcal{V}, c \oplus \beta, \mathcal{M})$ in which $\mathcal{V} = (\mathcal{T} + \mathcal{R})$ mod N, $c \oplus \beta \in \{0, 1\}$, and $\mathcal{M}$ is a ciphertext. Correspondingly, $VIEW_2 = (\mathcal{W})$ where $\mathcal{W}$ is a ciphertext.

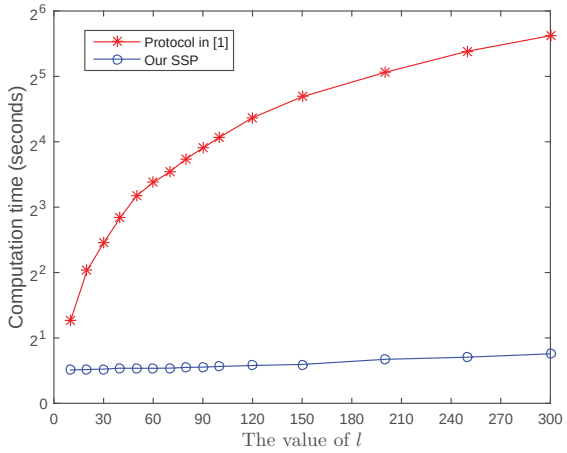
We can construct a simulator S_1 to simulate the client's view based on the following steps. When the client receives $\mathcal{V}$, S_1 randomly choose a number $\mathcal{V}^*$ from $\mathbb{Z}_N$. While the client gets $c \oplus \beta$, S_1 selects a random bit $c^* \in \{0, 1\}$. S_1 randomly chooses a random number $\mathcal{M}^*$ from $\mathbb{Z}_{N^2}$ when the client receives $\mathcal{M}$. The output of simulator S_1 is $S_1 = (\mathcal{V}^*, c^*, \mathcal{M}^*)$. Since Paillier encryption scheme is semantically secure, $\mathcal{M}^*$ is computationally indistinguishable from $\mathcal{M}$. As $\mathcal{R}$ and c are randomly chosen from $\mathbb{Z}_N$ and $\{0, 1\}$, $\mathcal{V}^*$ and c^* are computationally indistinguishable from $\mathcal{V}$ and $c \oplus \beta$ respectively. Based on the above analysis, we can conclude that S_1 is computationally indistinguishable from $VIEW_1$, which means equation (17) holds.

Meanwhile, we can construct a simulator S_2 to simulate the server's view. The output of simulator S_2 is $S_2 = (\llbracket b' \rrbracket)$ where b' is randomly chosen from $\mathbb{Z}_N$. Since Paillier encryption is semantically secure, $\llbracket b' \rrbracket$ is computationally indistinguishable from $\mathcal{W}$. As a result, S_2 is computationally indistinguishable from $VIEW_2$, which means equation (18) holds.

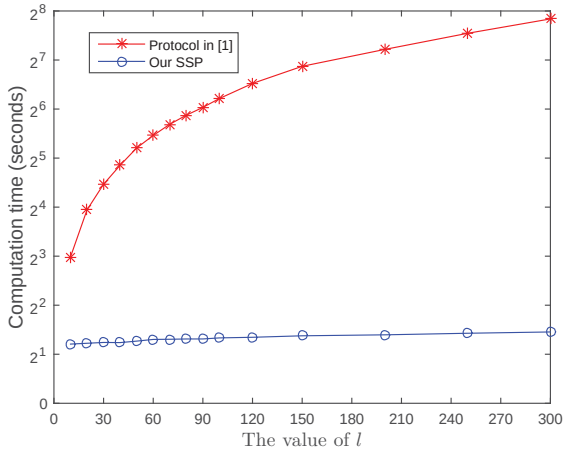
Additionally, the security of garbled circuits has been formally proved in [13]. In general, we can claim that our proposed SSP is secure under the semi-honest model. □

5.3 Cost Evaluation

Computation Cost. In our SSP, the server needs 3 encryptions and 5 ciphertext multiplications to compute $\mathcal{U}$, $\mathcal{T}$, $\mathcal{V}$ and $\mathcal{M}$. Besides, the server constructs the garbled circuit for comparison.



(a) The bit length of key is 1024.



(b) The bit length of key is 2048.

Fig. 2. Computation time comparison of Rahulamathavan *et al.*'s scheme [1] and our SSP.

The client implements 2 decryptions (to obtain $\mathcal{V}$ and M_c), 1 encryption (to obtain $\llbracket c \oplus \beta \oplus \mathcal{V}_0 \rrbracket$), and the garbled circuit evaluation (to obtain $c \oplus \beta$).

In our SSP, an improved comparison garbled circuit is executed for secure comparison. Considering the free-XOR technique [15], the computation cost of garbled circuits only depends on the number of non-XOR gates in the circuit. Using garbled circuits designed in [14], the number of non-XOR gates in comparison circuit with two x-bit as input is x. Thus, $\lceil \log N \rceil$ non-XOR gates are required in our approach, where $\mathbb{Z}_N$ is the plaintext space of Paillier encryption system and $\lceil \cdot \rceil$ represents ceiling function.

In total, our proposed scheme requires 4 encryptions, 5 ciphertext multiplications, 2 decryptions, and $\lceil \log N \rceil$ non-XOR gates.

Communication Overheads. Since the ciphertext space of Paillier encryption system is $\mathbb{Z}_{N^2}$, then the bit length of an encrypted number is $2 * \lceil \log N \rceil$. Thus, apart from garbled circuit protocol for plaintext com-

parison, our SSP only costs $6*\lceil \log N \rceil$ bits to transmit three encrypted values $[\![V]\!]$, $\mathcal{W}$ and $\mathcal{M}$. Let $\mathcal{C}$ be the communication overheads of garbled circuit protocol in line 4 of our SSP. Our scheme will totally require $(6*\lceil \log N \rceil + \mathcal{C})$ bits for communication.

Simulation Evaluation and Comparison. We further compare Rahulamathavan *et al.*'s scheme [1] and our SSP through simulation experiments. We implement both schemes using Java. The simulation environment for both schemes is Windows 10 with Intel Core i3-3220 3.30GHz CPU and 8.0GB memory. While implementing Rahulamathavan *et al.*'s scheme, secure comparison protocol in [3] is utilized as a basic building block to attain an encrypted comparison result of $(z_r \mod 10^l)$ and $(r \mod 10^l)$, according to the description in [1]. In both schemes, the involved Paillier encryption is achieved by employing Paillier library in [16], and the garbled circuit protocol is achieved by using the open-source FastGC [11].

As shown in Fig. 2, our proposed SSP is always faster than Rahulamathavan *et al.*'s scheme. With the growth of l, SSP can save more computation time. The reasons are as follows. Rahulamathavan *et al.*'s scheme obtains the encrypted comparison result $[\![\lambda]\!] = [\![(z_r \mod 10^l) < (r \mod 10^l)]\!]$ by employing the protocol in [3], [4], where each bit of $(r \mod 10^l)$ and $(z_r \mod 10^l)$ will be encrypted, and the comparison is performed over the encrypted bits. It is more time-consuming, while l is larger. In contrast, our SSP only involves 11 encryptions/decryptions, one garbled circuit protocol for comparing $\big((\mathcal{T} + \mathcal{R}) \mod N\big)$ and $(\mathcal{R} \mod N)$. It is easy to say that the size of the garbled circuit protocol in our solution scales linearly as the bit-length of N. While N is fixed, the size of the garbled circuit protocol in our scheme is constant for any l. Thus, our computation time increases, as the key is longer. Nevertheless, the value of l has little influence on our computation time, since the size of the garbled circuit protocol in our solution does not vary with l. For a fixed key, the computation time of SSP almost remains the same, as l varies. While $l = 300$ and the key is 1024-bit, the computation time of SSP is less than 2 seconds. While $l = 300$ and the key is 2048-bit, our SSP merely takes about 2.8 seconds. We also evaluate our scheme on a real dataset, "a1a" from LIBSVM data sets [12]. The results indicate while $p = 3$ to 5 and the scaling factor γ is set to 10^6, the computation time of our scheme to determine each encrypted number's sign is less than 3 seconds. Therefore, the proposed SSP is of high efficiency to securely determine the sign of $d(\boldsymbol{t})$ in encrypted domain.

6 Conclusion

In this paper, we analyzed the privacy preserving SVM classification scheme recently proposed in [1], and pointed out that their core protocol (i.e., a secure protocol to determine encrypted numbers' sign) suffers from soundness problem or security leakage. Then, we proposed a new approach to correctly and securely determine encrypted numbers' sign, which can be utilized to support privacy-preserving SVM classification. Theoretical analysis show our proposed protocol can fix all the flaws. Additionally, evaluation results indicate our scheme can achieve higher efficiency than the existing one in [1].

Acknowledgments

This work is partly supported by the Natural Science Foundation of China (No. 61602240), the Natural Science Foundation of Jiangsu Province of China (No. BK20150760), and the Research Fund of Guangxi Key Laboratory of Trusted Software (No. kx201611).

References

[1] Y. Rahulamathavan, R. C. W. Phan, S. Veluru, K. Cumanan, M. Rajarajan. Privacy-preserving multi-class support vector machine for outsourcing the data classification in cloud. IEEE Transactions on Dependable and Secure Computing, 2014, 11(5): 467-479.

[2] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. Advances in cryptology - EUROCRYPT, Springer Berlin Heidelberg, 1999: 223-238.

[3] I. Damgård, M. Geisler, M. Krøigaard. Efficient and secure comparison for on-line auctions. Proc. of the 12th Australasian Conference on Information Security and Privacy (ACISP), LNCS 4586, 2007: 416-430.

[4] J. Garay, B. Schoenmakers, J. Villegas. Practical and Secure Solutions for Integer Comparison. Proc. of the 10th International Conference on Practice and Theory in Public-Key Cryptography (PKC), LNCS 4450, 2007: 330-342.

[5] S. Tong, D. Koller. Support vector machine active learning with applications to text classification. Journal of Machine Learning Research, 2002, 2(1):45-66.

[6] N. Sapankevych, R. Sankar. Time series prediction using support vector machines: a survey. IEEE Computational Intelligence Magazine, 2009, 4(2):24-38.

[7] S. Hu, Y. Yao, Z. Yang. MAC protocol identification using support vector machines for cognitive radio networks. IEEE Wireless Communications, 2014, 21(1):52-60.

[8] M. Kang, J. Kim, J.M. Kim, A. Tan, E. Kim, B. Choi. Reliable Fault Diagnosis for Low-Speed Bearings Using Individually Trained Support Vector Machines With Kernel Discriminative Feature Analysis. IEEE Transactions on Power Electronics, 2015, 30(5):2786-2797.

[9] A. C. Yao. How to Generate and Exchange Secrets. Proc. of the 27th Annual Symposium on Foundations of Computer Science (FOCS) 1986: 162-167.

[10] O. Goldreich. Foundations of cryptography. Cambridge University Press, 2004.

[11] Y. Huang, D. Evans, J. Katz, L. Malka. Faster secure two-party computation using garbled circuits. in Usenix Security Symposium, 2011.

[12] https://www.csie.ntu.edu.tw/čjlin/libsvmtools/datasets/

[13] Lindell, Yehuda and Pinkas, Benny. A proof of security of Yaos protocol for two-party computation. Journal of Cryptology, 2009, 22(2): 161-188.

[14] Kolesnikov, Vladimir and Sadeghi, Ahmad-Reza and Schneider, Thomas. Improved garbled circuit building blocks and applications to auctions and computing minima. International Conference on Cryptology and Network Security, 2009: 1-20.

[15] Kolesnikov, Vladimir and Schneider, Thomas. Improved garbled circuit: Free XOR gates and applications. International Colloquium on Automata, Languages, and Programming, 2008:486-498.

[16] Liu, Kun and Bhaduri, Kanishka and Das, Kamalika and Nguyen, Phuong and Kargupta, Hillol. Client-side web mining for community formation in peer-to-peer environments. ACM SIGKDD Explorations Newsletter, 2006, 8(2): 11-20.