Lightweight Privacy-Preserving Ensemble Classification for Face Recognition

Zhuo Ma, Yang Liu, Ximeng Liu, *Member, IEEE*, Jianfeng Ma, and Kui Ren, *Fellow, IEEE*

***Abstract*—The development of machine learning technology and visual sensors is promoting the wider applications of face recognition into our daily life. However, if the face features in the servers are abused by the adversary, our privacy and wealth can be faced with great threat. Many security experts have pointed out that, by 3-D-printing technology, the adversary can utilize the leaked face feature data to masquerade others and break the E-bank accounts. Therefore, in this paper, we propose a lightweight privacy-preserving adaptive boosting (AdaBoost) classification framework for face recognition (POR) based on the additive secret sharing and edge computing. First, we improve the current additive secret sharing-based exponentiation and logarithm functions by expanding the effective input range. Then, by utilizing the protocols, two edge servers are deployed to cooperatively complete the ensemble classification of AdaBoost for face recognition. The application of edge computing ensures the efficiency and robustness of POR. Furthermore, we prove the correctness and security of our protocols by theoretic analysis. And experiment results show that, POR can reduce about 58% computation error compared with the existing differential privacy-based framework.**

***Index Terms*—Adaptive boosting (AdaBoost), additive secret sharing, face recognition, privacy-preserving.**

I. Introduction

ALONG with the development of machine learning, face recognition has become the second popular biometric authentication technology next to the fingerprint. Due to the security and convenience, people are enjoyable to choose the face feature-based biometric authentication on protecting their property. For example, as early as 2013, PayPal provides the service of checking in and paying with face for more than 22 million users in England [1]. And in 2014, the presentation of hierarchical learning architecture for face recognition leads to the accuracy reaching to more than 99.5%, which further guarantees the dependability and practicality of the face recognition service [2]. Although the performance of machine learning-based face recognition technology has far exceeded traditional methods [3], it is still a consensus that the memory space and computational power requirement for its training process is quite high. For instance, in a large-scale face recognition dataset for people of different ages, there are 3.31 million images of 9131 subjects that the machine learning model has to learn [4]. To lower the cost and satisfy the high demand for computing capability, many app operators prefer to outsource their intensive computation and extreme volume of data to the cloud server [5]. Nevertheless, for most of the face recognition applications, it is required that thousands of users can get the feedback in seconds. Since the cloud servers are usually far away from the service request location, meeting the demand for such degree of latency may lead to high pressure on data communication speed and robustness. Thus, a new paradigm edge computing is adopted to solve this problem. Edge computing is one of the hot spots that arises dramatic interests of industry investment and research [6]. By placing the computing and storage center at the Internet's edge close to image collection device, the paradigm can effectively decrease the communication latency and strengthen the robustness [7].

However, due to the lack of privacy protection in most practical applications, the face feature data for machine learning are usually outsourced to servers without encryption and at the risk of being eavesdropped and abused [8]. Once the adversaries do achieve their goals, it may not only lead to the infringement of portrait rights, but also the severe damage to the property of users. More seriously, the face recognition surveillance in public spaces has arisen the social debate about the invitation to personal privacy [9]. Many people complain that they walk through a market just like going through a reticent trial, because their faces are ceaselessly matched against a government gallery of culprits by the ubiquitous cameras and they even do not know, where these "inquisitors" are. Thus, a novel face recognition framework is needed to undertake the recognition task without disclosure of personal face features.

For privacy preservation of ensemble learning, the most common way is to use the differential privacy (DP) technology. Wang *et al.* [10] chose the DP-based deep learning framework to implement the client privacy of detecting credit card fraud. However, due to the addition of Laplace distribution or Gaussian distribution-based random noise, the

Manuscript received January 28, 2019; revised March 9, 2019; accepted March 13, 2019. Date of publication March 18, 2019; date of current version June 19, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant U1764263, Grant 61872283, Grant 61702105, and Grant U1804263, in part by the Natural Science Basic Research Plan in the Shaanxi Province of China under Grant 2016JM6074, and in part by the China 111 Project under Grant B16037. *(Zhuo Ma, Yang Liu, and Ximeng Liu contributed equally to this work.) (Corresponding author: Yang Liu.)*

Z. Ma, Y. Liu, and J. Ma are with the School of Cyber Engineering, Xidian University, Xi'an 710071, China (e-mail: mazhuo@mail.xidian.edu.cn; bcds2018@foxmail.com; jfma@mail.xidian.edu.cn).

X. Liu is with the School of Information Systems, Singapore Management University, Singapore, and also with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China (e-mail: snbnix@gmail.com).

K. Ren is with the Institute of Cyberspace Research, Zhejiang University, Zhejiang, China (e-mail: kuiren@zju.edu.cn).

Digital Object Identifier 10.1109/JIOT.2019.2905555

stronger privacy we want to obtain, the greater computation error the DP would introduce [11]. The other method to preserve privacy that does not cause such great error is homomorphic encryption (HE). In recent years, plenty of HE-based frameworks for machine learning are proposed, like the SecureML designed by Mohassel and Zhang [12] and the DeepZeroID [13]. Although great interests are put on HE, many people point that the nature of high time-complexity and intensive memory consumption makes it unpractical in real-world applications. Accordingly, we now need a errorless and efficient framework to address the privacy problem of face recognition.

For this purpose, we propose a lightweight privacy-preserving adaptive boosting (AdaBoost) face recognition framework (POR) based on additive secret sharing secret. In POR, all of the face features and pivotal machine learning parameter are computed in the encrypted form. However, the previously proposed secure exponentiation and logarithm functions are not additive or cannot converge effectively when the inputs are not between 0 and 1 [14], which can cause them unusable in most machine learning-based applications. Therefore, by utilizing the single-precision floating-point number representation, we expand the effective input ranges to arbitrary input range. Then, a series of interactive protocols are designed to efficiently complete the computation of the forward stagewise algorithm (FSW) and linear addition process of AdaBoost. By calling the basic linear secure functions and Maclaurin Series to approximate the exponentiation and logarithm functions, the protocols are efficient in computation and can cause controllable error. Moreover, because of narrowing the distance between the image acquisition device and the cloud servers, edge computing is also one of the keys to make our framework more practical in application. Specially, the basic classifier is alterable in POR, but it has to be privacy-preserving. The contributions of this paper can be summarized as follows.

1) We propose an AdaBoost-based framework POR for protecting the privacy of both the user face features and the service provider's pivotal learning parameters in ensemble neural network. By adopting the additive secret sharing and the edge computing technology, POR is more efficient and less error-prone.
2) We improve the existing additive interactive protocols for exponentiation and logarithm functions by expand the effective input range from 0 to 1 to arbitrary inputs.
3) A series of interactive protocols are specially designed for the different training stage of AdaBoost. The protocols ensure that the data in the forward stagewise process and linear addition of weak classifiers are computed in the encrypted form.
4) By experiments, we further prove the efficiency and accuracy of POR. The results show that POR can reduce about 58% computation error compared with the DP-based framework, and the additional cost of the improved secure nonlinear functions is only a few millisecond computing time and no more than 500 bit communication overhead.

II. Preliminaries

In this section, some basic information about the AdaBoost and several secret sharing-based mathematical functions are introduced.

A. AdaBoost

AdaBoost is a boosting classifier which is able to combine multiple weaker classifiers into a stronger one. In the training process, the weight vector ω of AdaBoost is successively updated based on the error rate of weak classifiers. The final output is the target function $f(x)$ obtained from the cumulative weighted prediction results of per training round. Given the training dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, we can mathematically express the whole training process as follows.

1) Initiate the weight vector ω before iteration

$$\omega_1 = \{\omega_{1,1}, \omega_{1,2}, \ldots, \omega_{1,N}\} \quad \omega_{1,i} = \frac{1}{N}. \tag{1}$$

2) Suppose the weak classifier is $C_k(x)$, where k is the number of iteration round. The error rate e_k of iteration k is

$$e_k = \sum_{i=1}^{N} \omega_{k,i} \cdot I(x_i), \quad i = 1, 2, \ldots, N \tag{2}$$

$$I(x_i) = \begin{cases} 0, & \text{if } C(x_i) = y_i \\ 1, & \text{if } C(x_i) \neq y_i. \end{cases} \tag{3}$$

3) Then, the impact factor of $C_k(x)$ in the final output is computed

$$\lambda_k = \frac{1}{2} \ln \frac{1 - e_k}{e_k}. \tag{4}$$

4) With λ_k, the values of ω can be can be updated as

$$\omega_{k+1} = \frac{\omega_{k,i}}{\sigma_k} e^{-\lambda_k y_i C_k(x_i)} \tag{5}$$

$$\sigma_k = \sum_{i=1}^{N} \omega_{k,i} e^{-\lambda_k y_i C_k(x_i)}. \tag{6}$$

5) When the termination condition like reach the maximum iteration number is satisfied, output $O(x)$

$$f(x) = \text{sign}\left(\sum_{i=1}^{\infty} \lambda_i C_i(x)\right) \tag{7}$$

$$\text{sign}(x) = \begin{cases} -1, & \text{if } x < 0 \\ 1, & \text{if } x > 0. \end{cases} \tag{8}$$

B. Secret Sharing-Based Mathematical Functions

To maintain the security and privacy for machine learning models, the normal mathematical functions are widely substituted by secret sharing-based ones in our framework. For convenience of later discussion, we would like to list some previously proposed protocols [15]–[17]. All these protocols run on two edge servers with a trusted third server to generate uniformly random values. The inputs u and v are the data required to be securely computed.

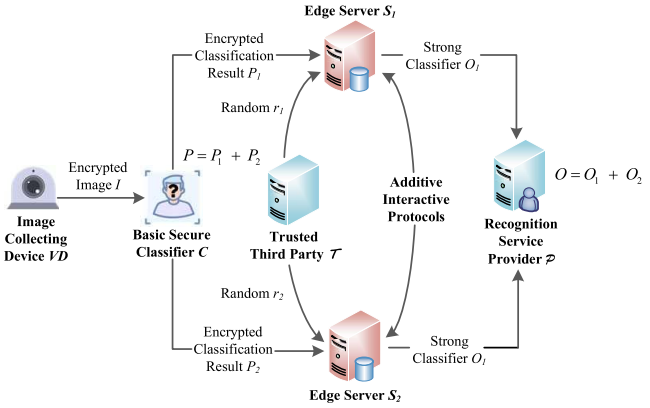

Fig. 1. System model.

1) *Random Bits Protocol:* The *RanBits*$(\cdot)$ dose not have input and generates arbitrary length of uniformly random bit sequence $(r_0, \ldots, r_\ell)$ used for the other protocols.
2) *Secure Comparison Protocol:* Set the inputs to be u and v. By invoking the protocol SecCmp$(\cdot)$, S_1 and S_2 can judge which one is larger. If $u > v$, SecCmp$(\cdot)$ outputs 1; if $u < v$, it outputs -1; otherwise, it outputs 0.
3) *Secure Multiplication Protocol:* Given the inputs u and v, the secure multiplication protocol SecMul$(\cdot)$ masks them with a *Beaver's triplet* (a, b, c). And then, the two edge servers can compute $f = u \cdot v$ without disclosure of any information about the inputs. To recover the final computation result, we just have to collect the secret shares and compute $f = f_1 + f_2$.
4) *Secure Reciprocal Protocol:* Set the input be u. The Newton–Raphson iterative method base reciprocal protocol SecInv$(\cdot)$ iteratively approximate to the real result $f = (u/v)$. When the precision is satisfies, the iteration terminates and SecInv$(\cdot)$ outputs $f = f_1 + f_2 \approx (u/v)$.
5) *Secure Natural Exponential Protocol:* Suppose the input is u, the Maclaurin Series-based protocol SecExp$(\cdot)$ outputs $f_1 + f_2 = e^u$. During computation, the two edge servers iteratively compute the series to approximate the real result. The input of SecExp$(\cdot)$ must be in the range of 0 and 1 to make the series converge, which makes it inapplicable in most applications.
6) *Secure Natural Logarithm Protocol:* The protocol SecLog$(\cdot)$ is based on the modified Maclaurin Series which by far converges the fastest. However, similar to SecExp$(\cdot)$, its input is also limited between 0 and 1. Given an input u, SecLog$(\cdot)$ outputs $f_1 + f_2 = \ln u$.

III. System Model and Attack Model

A. System Model

As illustrated in Fig. 1, the proposed privacy-preserving face recognition system comprises five types of participants, which are image collecting device (*VD*), basic secure classifier (C), edge servers (S_i, $i \in \{1, 2\}$), trusted third party ($\mathcal{T}$), and recognition service provider ($\mathcal{P}$).

1) The *VD* can be any IoT device that can operate a visual task, such as a smartphone or a camera that is connected to the Internet. The face images are encrypted here, and then sent to the server which deploys C.
2) The C is a weak privacy-preserving face recognizer, like the secure support vector machine [18]. It is able to do the preliminary classification work and outputs result $P = P_1 + P_2$ to S_i, but its accuracy is far more lower than the actual applications require.
3) There are two edge servers S_1 and S_2 in our system model which are dedicated to doing the complex computation of POR. All of the interactive protocols designed in this paper are operated between them. And they get the encrypted dataset D in additive secret sharing mode from C as inputs in the training process of POR. If necessary, the two edge servers can also possess the capacity to generate uniformly random values.
4) Since the work of $\mathcal{T}$ is only responsible for generating random values r_i to mask secret shares of S_i, a lightweight server is deployed as $\mathcal{T}$.
5) The $\mathcal{P}$ can simply get the final strong classifier by computing $O = O_1 + O_2$. And if the training result is appropriate, (s)he can then deploy O_i on S_i for his commercial applications or other purposes.

Note that the system model is two-party setting (2PC) which is the same as the preliminary protocols in Section II. If multiple edge servers need to be introduced, the only thing need to do is changing SecCmp$(\cdot)$ and SecMul$(\cdot)$ into multi-party setting (MPC), because the other three related functions in Section II are all iterative series-based, and the series are only composed of the two functions. From [15] and [16], it can be found that the changes for SecCmp$(\cdot)$ and SecMul$(\cdot)$ are completely realizable.

B. Attack Model

We assume all of participants in POR are *honest-but-curious*. Normally, they receive the messages and response them fully in accordance with the interactive protocols. But once given opportunity, they do not refuse to learn anything that is beneficial to themselves. Furthermore, there are two simulators π_1 and π_2 that can simulate the protocol operation process between S_1 and S_2. Both of them have the following abilities: 1) π_1 and π_2 are polynomial-time simulators; 2) π_1 and π_2 can generate uniformly random values; 3) according to the real view of the protocol, π_1 and π_2 can simulate the whole protocol operation process; and 4) the set Sim_i stores all of the values simulated by π_i about the protocol operation process.

Furthermore, we hypothesize that the edge servers can learning nothing but the messages sent by the other participant in accord with the protocol and cannot collude with each other. *VD*, C, $\mathcal{T}$, and $\mathcal{P}$ are all honest. And there are secure communication channels among the honest participants. A successful attack in our model means that Sim_i and the real view of the protocol are computationally distinguishable for the adversary $\mathcal{A}$.

IV. Secret Sharing-Based Functions

In this section, several secure interactive subprotocols are designed to implement the privacy-preserving AdaBoost. Compared with the previously proposed protocols [16], the improved secure natural exponentiation ISExp(·) and the secure natural logarithm ISLog(·) do not have to limit the range of input. The improvement is seriously important, because, in AdaBoost, there is no sigmoid function or tanh function to keep the intermediate value of computation between 0 and 1 for maintaining the effectiveness of Maclaurin Series. In addition, for brevity, the notion x_i is used to represent the shares possessed by S_i, where $i \in \{1, 2\}$.

A. Data Storage Format

To overcome the problem that the series converge too slow when the inputs are far away from 0, we adopt the single-precision floating-point number representation to further improve the secure logarithm function. The representation conversion can be directly operated on the image acquisition device or locally computed in the cloud servers. Moreover, PPOR can also support the double-precision representation, but 32-bit length single-precision secret shares are secure enough for most applications. Accordingly, only the single-precision data representation method is presented here. If required, all of the protocols in our scheme can be easily expanded into double-precision mode by increasing the bit-length of mantissa m and exponent ε. The cost incurred by the expansion is just additional communication overhead.

1) Data Representation: The single-precision float-point representation we adopt is based on the standard format deployed in IEEE 754-1985, which is also called binary32 in IEEE 754-2008 [19]. To adapt to our need, several changes on the original representation are made in our scheme. First, the number is not stored into a fixed 32-bit storage space, but one 32-bit float m and one 16-bit integer ε which are, respectively, corresponding to the mantissa and the exponent. The sign bit for standard format is served. The bias for exponent is set to 0, because, for our protocols, it can only cause extra computations. Above all, for an arbitrary signed number u, it can be converted into the form

$$u = m \cdot 2^{e-\text{bias}} \quad \text{bias} \in \{0, 1, 2, \ldots, 255\} \tag{9}$$

where $-1 \leq m \leq 1$ and ε is a signed integer.

B. Secure Matching of Exponents

With the single-precision type of representation, the exponents of the secret shares to be calculated may not match. Therefore, one of the most important premise for the secure function computation is matching the exponents of the two inputs. As shown in Algorithm 2, given two secret shares the secure matching of exponents protocol SME(·) makes sure them have same exponents by negotiation. During the process, the two participants do not expose their own sharing to each other. And sine the data representation method do not change the values of the original inputs, SME(·) introduce no extra computation errors into the system. In addition, if the data format is extended to double-precision, SME(·) can still work. According to the standard definition of double-precision, its difference with single-precision is the valid bits lengths of mantissa and exponent. The difference is trivial for SME(·) and cannot influence the validity of the protocol.

Algorithm 1 Format Conversion of Single-Precision

Input: Secret share x;
Output: Mantissa m and exponent ε;

```
SINGLE-PRECISION(x)
    m ← x
    ε ← 0
    while m > 1 or m < −1 do
        m ← m/2
        ε ← ε + 1
    end while
    return m, ε
end
```

Algorithm 2 Secure Matching of Exponent (SME)

Input: S_1 has input u_1; S_2 has input u_2;
Output: S_1 outputs u_1; S_2 outputs u_2; S_1 and S_2 both output ε;

1: S_1 has $m_1, \varepsilon_1 \leftarrow$ SINGLE-PRECISION(u_1).
2: S_2 has $m_2, \varepsilon_2 \leftarrow$ SINGLE-PRECISION(u_2).
3: S_1 splits m_1 into $m_1 \leftarrow m_1' + m_1''$ and computes $u_1' = m_1' \cdot 2^{\varepsilon_1}$, $u_1'' = m_1'' \cdot 2^{\varepsilon_1}$.
4: S_2 splits m_2 into $m_2 \leftarrow m_2' + m_2''$ and computes $u_2' = m_2' \cdot 2^{\varepsilon_2}$, $u_2'' = m_2'' \cdot 2^{\varepsilon_2}$.
5: S_1 sends u_1'' to S_2 and S_2 sends u_2' to S_1
6: S_1 locally matches the exponent of u_1' and u_2' by computing $\varepsilon \leftarrow max(\varepsilon_1, \varepsilon_2)$, then updates u_1 by calculating $u_1 \leftarrow u_1' + u_2'$.
7: S_2 locally matches the exponent of u_1'' and u_2'' by computing $\varepsilon \leftarrow max(\varepsilon_1, \varepsilon_2)$, then updates u_2 by calculating $u_2 \leftarrow u_1'' + u_2''$.
8: S_i returns u_i and ε

1) Secure Matching of Exponent: To make the two shared numbers have the same exponents, S_1 and S_2 first have to convert their inputs into single-precision format $u_1 = m_1 \cdot 2^{\varepsilon_1}$ and $u_2 = m_2 \cdot 2^{\varepsilon_2}$ by invoking the Algorithm 1, the format conversion function SINGLE-PRECISION(·). As mentioned before, the format conversion has been completed in advance. Then they split the inputs into secret shares $u_1 = u_1' + u_1'' = m_1' \cdot 2^{\varepsilon_1} + m_1'' \cdot 2^{\varepsilon_1}$ and $u_2 = u_2' + u_2'' = m_2' \cdot 2^{\varepsilon_2} + m_2'' \cdot 2^{\varepsilon_2}$, where $m_1 = m_1' + m_1''$ and $m_2 = m_2' + m_2''$. Also, we have $m_i' \neq 0$ and $m_i'' \neq 0$. Next, m_i' and m_i'' are, respectively, distributed to S_1 and S_2. Locally, S_1 updates the value of u_1 by computing $u_1 = u_1' + u_2'$, and S_2 sets $u_2 = u_1'' + u_2''$. It can be derived that, to complete the computation, S_i have matched the exponents of the two secret shares, and we still have $u = u_1 + u_2$. Thus, the inputs are securely converted into single-precision floats with same exponents. Specially, no matter the sum of mantissas is more than 1 or not, the final exponent is not influenced and always the larger one of ε_1 and ε_2.

Algorithm 3 Improved Secure Natural Exponential Function (ISExp)

Input: S_1 has input u_1; S_2 has input u_2; The required precision is ϵ

Output: S_1 outputs f_1; S_2 outputs f_2;

1: S_1 computes $u_1 \leftarrow \alpha_1 + \beta_1$, where α_1 is integer and $0 \leq \beta_1 < 1$.
2: S_2 computes $u_2 \leftarrow \alpha_2 + \beta_2$, where α_2 is integer and $0 \leq \beta_2 < 1$.
3: S_1 and S_2 compute $v_1, v_2 \leftarrow SecExp(\beta, \epsilon)$, where $\beta \leftarrow \beta_1 + \beta_2$ and $v \leftarrow v_1 + v_2$.
4: S_1 computes $a \leftarrow e^{\alpha_1}$ and splits it into random shares $a \leftarrow a_1 + a_2$
5: S_1 computes $b \leftarrow e^{\alpha_2}$ and splits it into random shares $b \leftarrow b_1 + b_2$
6: S_1 sends a_2 to S_2 and S_2 sends b_1 to S_1
7: S_1 and S_2 compute $(p_1, p_2) \leftarrow SecMul(a, b)$
8: S_1 and S_2 compute $(f_1, f_2) \leftarrow SecMul(p, v)$
9: S_i returns f_i

Algorithm 4 Improved Secure Natural Logarithm Function (ISLog)

Input: S_1 has input u_1; S_2 has input u_2; The required precision is ϵ

Output: S_1 outputs f_1; S_2 outputs f_2;

1: S_1 computes $m_1, \varepsilon \leftarrow \text{SME}(u_1)$.
2: S_2 computes $m_2, \varepsilon \leftarrow \text{SME}(u_2)$.
3: S_1 and S_2 compute $v_1, v_2 \leftarrow SecLog(m, \epsilon)$, where $m \leftarrow m_1 + m_2$ and $v \leftarrow v_1 + v_2$.
4: S_1 computes $f_1 \leftarrow v_1 + \varepsilon \cdot \ln 2$
5: S_2 computes $p_1 \leftarrow v_2$
6: S_i returns f_i

C. Improved Secure Natural Exponential Function

Given an encrypted input u, ISExp(·) can always converge at ideal speed and output the natural exponentiation $f(u) = e^u$. The inputs of ISExp(·) support 32-bit and 64-bit floating-point types, which is the same the exponentiation function in almost every microcontroller and computer. As illustrated in Algorithm 3, compared with previously proposed SecExp(·), ISExp(·) does not limit the range of input value by virtue of splitting the input into additive integer part and fractional part. And to achieve the improvement, what is extra brought by our method is just two times of secure multiplications and two locally computed exponents. In addition, even the effective range is not limited, it is better to keep the inputs no more than 10^2. Because if not, even for the frameworks without privacy-preserving, it may cause intolerable huge calculation for large-scale datasets. Thus, the actual average computation time of ISExp(·) in application is actually less the theoretical analysis result.

Initialization: S_i gets its input as u_i which satisfies $u = u_1 + u_2$. Set the precision be $\epsilon = \epsilon_1 + \epsilon_2$ and distribute ϵ_i to S_i. ϵ is used for judging when to terminate the iteration of SecExp(·).

Secure Natural Exponentiation: S_1 and S_2, respectively, split their inputs into integer part and fractional part $u_i = \alpha_i + \beta_i$, where $0 \leq \beta_i < 1$. Then, the edge servers utilize the old secure natural exponential function to compute $(v_1, v_2) \leftarrow \text{SecExp}(\beta, \epsilon)$, where $v = v_1 + v_2$. Meanwhile, S_1 and S_2 locally compute $a = e^{\alpha_1}$ and $b = e^{\alpha_2}$. a and b are split into random shares $a = a_1 + a_2$ and $b = b_1 + b_2$. a_i and b_i are then distributed to S_i. By twice invoking the secure multiplication function, S_1 and S_2 continue to calculate $(p_1, p_2) \leftarrow \text{SecMul}(a, b)$ and output $(f_1, f_2) \leftarrow \text{SecMul}(p, v)$, where $p = p_1 + p_2$ and $f = f_1 + f_2$. The entire computation process can be mathematically expressed as

$$f(u) = e^u = e^{\alpha+\beta} = \left(\sum_{i=0}^{\infty} \frac{1}{i!} \cdot \alpha^i\right) \cdot e^{\beta}. \tag{10}$$

D. Improved Secure Natural Logarithm Function

1) Secure Natural Logarithm Function: Similar to the improved secure natural exponential function, ISLog(·) also gets rid of the input limit with the help of data format conversion. As shown in Algorithm 4, for an arbitrary input u, what we should compute in ISLog(·) becomes $f(u) = \ln(m \cdot 2^{\varepsilon}) = \ln m + \ln 2^{\varepsilon}$, where $0 < m < 1$. SecLog(·) can be directly invoked to compute $\ln m$, because the series used in SecLog(·) is by far the fastest to converge in this range of input. And to get the final result, let S_1 or S_2 locally do one multiplication and one addition. Furthermore, it can be observed that, compared with the original secure function, ISLog(·) do not introduce any extra error.

Initialization: The initialization process of ISLog(·) is the same as ISExp(·).

Secure Natural Logarithm: During online computation phase, S_1 and S_2 first have to convert the inputs into our single-precision format. To achieve this, the edge servers update the values of u_1 and u_2 by invoking SME(u). Then, the original secure natural logarithm function is used to calculate $(v_1, v_2) \leftarrow \text{SecLog}(m)$. And S_1 and S_2, respectively, compute $f_1 = v_1 + \varepsilon \cdot \ln 2$ and $f_2 = v_2$. the final outputs f satisfy $f = f_1 + f_2$. The details of computation process can also be expressed as mathematical expression

$$f(u) = \ln u = \ln\left(m \cdot 2^{\varepsilon}\right) = \ln m + \varepsilon \cdot \ln 2 = 2 \cdot \sum_{i=0}^{\infty} \frac{1}{2i+1} \cdot m^{2i+1} + \varepsilon \cdot \ln 2. \tag{11}$$

V. Privacy-Preserving AdaBoost-Based Face Recognition

In this section, we design a privacy-preserving framework for the training process of AdaBoost. The framework is composed of two parts: 1) the secure addition model and 2) the secure FSA. For the former part, the linear addition of a series of weak classifiers is computed, which can be locally completed by the secure addition function. As shown in Fig. 2, the FSA is the basis of the iterative training process of AdaBoost, and the secure functions proposed in Section IV are mainly designed for it. Specially, the weak classifier can be any privacy-preserving basic classifier, like the SVM in [18] and [20] and decision tree in [21]. Thus, we just

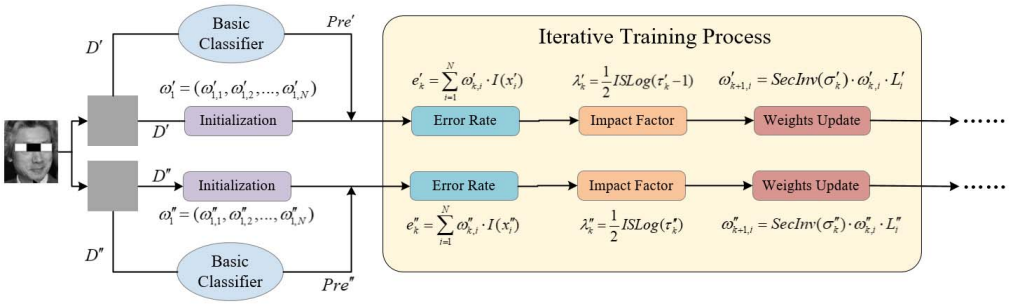

Fig. 2. Privacy-preserving FSW of AdaBoost.

utilize the notion $C'_k(x)$ and $C''_k(x)$ to denote the secret shares of iteration k classification results. Moreover, note that the multiplications of two secret shares in the remaining article are all computed by invoking the secure multiplication function SecMul$(\cdot)$.

A. Forward Stagewise Algorithm of AdaBoost

To get final strong classifier, AdaBoost deploys the FSA to iteratively strengthen the target classifier. The output of each iteration round is not independent but based on the previous training result, which can be simply expressed as

$$O_k(x) = O_{x-1}(x) + \lambda_k C_k(x; \alpha_k) \tag{12}$$

where α_k is the optimal parameter of $C_k(x)$. From (12), the FSA in our scheme is able to be concluded as computing the proportionality coefficient λ_k without disclosure of privacy. And the computation process is composed of the following four steps.

1) Initialization of Weight Vector: The first step of FSA is to initialize the weight vector $\omega_1 = (\omega_{1,1}, \omega_{1,2}, \ldots, \omega_{1,N})$. Set N be the number of samples. Let the trusted third party $\mathcal{T}$ generate two random shares vectors

$$\omega' = \left(\omega'_{1,1}, \omega'_{1,2}, \ldots, \omega'_{1,N}\right) \tag{13}$$

and

$$\omega'' = \left(\omega''_{1,1}, \omega''_{1,2}, \ldots, \omega''_{1,N}\right). \tag{14}$$

The elements of ω'_1 and ω''_1 satisfy $\omega'_{1,i} + \omega''_{1,i} = (1/N)$, where $i = \{1, 2, \ldots, N\}$ and $\omega_1 = \omega'_1 + \omega''_1$. It can be observed that ω_1 is actually publicly known. However, due to the fact that $\omega_k (k > 1)$ is private and required for our secure functions as inputs, we still use random shares to initialize them.

2) Error Rate: Suppose that the weak classifier $C_k(x)$ in iteration round k has been trained and its output is $\vec{P} = (p_{k,1}, p_{k,2}, \ldots, p_{k,N})$, where $p_{k,i} \in \{-1, 1\}$. Denote the training dataset as $D = \{(x_1, y_1) = (x'_1 + x''_1, y'_1 + y''_1), \ldots, (x'_N + x''_N, y'_N + y''_N)\}$, where $y_i \in \{-1, 1\}$. (x'_i, y'_i) and (x''_i, y''_i) are, respectively, possessed by S_1 and S_2. Then, let S_1 calculate

$$e'_k = \sum_{i=1}^{N} \omega'_{k,i} \cdot I\left(y'_i - p'_{k,i}\right) \tag{15}$$

and S_2 calculate

$$e''_k = \sum_{i=1}^{N} \omega''_{k,i} \cdot I\left(y''_i - p''_{k,i}\right). \tag{16}$$

Among them, $I(x_i)$ is implemented by invoking the secure comparison function SecCmp$(\cdot)$ which is an online protocol. Let S_1 and S_2, respectively, have the secret shares of $C_k(x)$ output $\vec{P}'_k = (p'_{k,1}, p'_{k,2}, \ldots, p'_{k,N})$ and $\vec{P}''_k = (p''_{k,1}, p''_{k,2}, \ldots, p''_{k,N})$, where $\vec{P}_k = \vec{P}'_k + \vec{P}''_k$. They first compute $(s'_1, \ldots, s'_N) = D.y' - \vec{P}'_k = (y'_1 - p'_{k,1}, \ldots, y'_N - p'_{k,N})$ and $(s''_1, \ldots, s''_N) = D.y'' - \vec{P}''_k = (y''_1 - p''_{k,1}, \ldots, y''_N - p''_{k,N})$. Then, judge whether $s_i = s'_i + s''_i$ is 0. If do, $I(s_i) = 0$, and otherwise, $I(s_i) = 1$. The process can also be expressed as

$$I\left(s'_i\right) = \begin{cases} 0, & \text{if SecCmp}\left(s'_i, 0\right) = 0 \\ 1, & \text{if SecCmp}(s_i, 0) \neq 0 \end{cases} \tag{17}$$

and

$$I\left(s''_i\right) = \begin{cases} 0, & \text{if SecCmp}\left(s'_i, 0\right) = 0 \\ 1, & \text{if SecCmp}\left(s'_i, 0\right) \neq 0. \end{cases} \tag{18}$$

3) Impact Factor: The impact actor λ_k determines how much impact each weak classifier $C_k(x)$ has on the final output. To obtain it, let the two edge servers cooperatively compute the reciprocal of the error rate e_k

$$\frac{1}{e_k} = \tau'_k + \tau''_k = \text{SecInv}\left(e'_k, e''_k\right). \tag{19}$$

Then, λ_k can be computed by invoking ISLog$(\cdot)$. Let S_1 have one share of the impact factor

$$\lambda'_k = \frac{1}{2}\text{ISLog}\left(\tau'_k - 1\right). \tag{20}$$

And S_2 has the other share of impact factor

$$\lambda''_k = \frac{1}{2}\text{ISLog}\left(\tau''_k\right). \tag{21}$$

4) Update of Weight Distribution: Before updating the weight distribution ω_k, the edge servers first have to compute the normalization factor σ_k. It guarantees that the probability distribution of dataset samples has a sum of 1. For efficiency, the intermediate results of loss L_i to compute ω_k are restored by the two edge servers and destroyed until one round of iteration is completed. By calling the improved secure exponential function, S_1 computes

$$\sigma'_k = \sum_{i=1}^{N} \omega'_k \cdot L'_i \tag{22}$$

$$L'_i = \text{ISExp}\left(-\lambda'_k \cdot y'_i \cdot C_k\left(x'_i\right)\right). \tag{23}$$

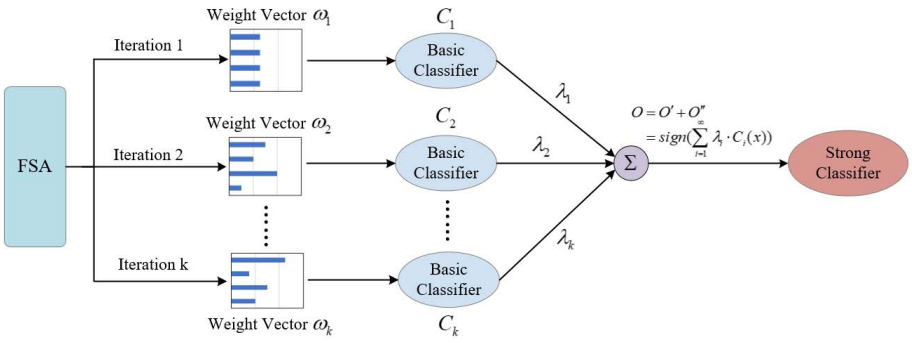

Fig. 3. Privacy-preserving linear addition of weak classifiers.

Meanwhile, S_2 computes

$$\sigma_k'' = \sum_{i=1}^{N} \omega_k'' \cdot L_i'' \tag{24}$$

$$L_i'' = \mathrm{ISExp}\left(-\lambda_k'' \cdot y_i'' \cdot C_k\left(x_i''\right)\right). \tag{25}$$

Then, S_1 and S_2 can, respectively, update the shares of weight vector by computing

$$w_{k+1,i}' = \mathrm{SecInv}\left(\sigma_k'\right) \cdot \omega_{k,i}' \cdot L_i' \tag{26}$$

and

$$w_{k+1,i}'' = \mathrm{SecInv}\left(\sigma_k''\right) \cdot \omega_{k,i}'' \cdot L_i''. \tag{27}$$

From the above four equations, the privacy-preserving AdaBoost adopts the exponential function as its loss function. As shown in Fig. 3, during the process of updating the weight distribution, what AdaBoost concentrates on are actually the samples that are not correctly recognized. The weights of misclassified samples are successively increased until they are be correctly classified. Oppositely, the weights of correctly recognized sample are decreased.

B. Linear Addition of Weak Classifier

As shown in Fig. 3, when the iterative training process of AdaBoost is terminated, we simply use linear addition of weak classifiers to get the final strong classifier. Before training termination, the weights of each samples are continually updated according the classification result. The misclassified samples are paid more concerns and their weights will be increased per iteration. On the contrary, the weights for correctly classified ones will be decreased. And the termination condition which can be threshold of accuracy or the number iteration round is determined by the service provider $\mathcal{P}$. Locally, S_1 and S_2, respectively, compute

$$O\left(x'\right) = \mathrm{sign}\left(\sum_{i=1}^{\infty} \lambda_i' \cdot C_i\left(x_i'\right)\right) \tag{28}$$

and

$$O\left(x''\right) = \mathrm{sign}\left(\sum_{i=1}^{\infty} \lambda_i'' \cdot C_i\left(x_i''\right)\right). \tag{29}$$

In (28) and (29), $\mathrm{sign}(\cdot)$ is a demodulation function. If the inputs is more than 0, $\mathrm{sign}(\cdot)$ outputs 1, otherwise outputs -1. By applying the secure comparison function, the two edge servers can work together to compute

$$O\left(x_i'\right) + O\left(x_i''\right) = \mathrm{sign}(x_i) = \begin{cases} 1, & \text{if } \mathrm{SecCmp}(x_i, 0) = 0 \text{ or } 1 \\ -1, & \text{if } \mathrm{SecCmp}(x_i, 0) = -1. \end{cases} \tag{30}$$

Thus, after operating all of the interactive protocols about the AdaBoost training process, S_1 and S_2 take the two secret shares O' and O'' of the strong classifier. If the accuracy of the final classifier reaches the degree that $\mathcal{P}$ requires, $\mathcal{P}$ is able to decrypt it by simply computing $O = O' + O''$. And given the test dataset $Q = \{(u_1, v_1) = (u_1' + u_1'', v_1' + v_1''), \ldots, (u_R' + u_R'', v_R' + v_R'')\}$, the test accuracy can be obtained by calling S_1 and S_2 to compute

$$\mathrm{Accuracy}_T = \frac{\sum_{i=1}^{R} I(v_i - p_i)}{R} \times 100\%. \tag{31}$$

Furthermore, the privacy-preserving AdaBoost is a universal ensemble learning model for most image recognition task. Consequently, if necessary, the weak classifier in our framework can be substituted by any other basic classifier, like cart decision tree or Fisher linear discriminant without influence of the privacy-preserving feature.

C. Extension for Multiclassification

The original AdaBoost is a binary classification model which is not appropriate for some face recognition tasks. Therefore, we also implement a variant of privacy-preserving Adboost called AdaBoost.M1. The variant dose not change the influence of the weak classifier on final output, but does a little adjustment for the weight update process.

First, the impact actor λ_k becomes

$$\lambda_k' = \mathrm{SecInv}\left(1 - e_k'\right) - 1 \tag{32}$$

and

$$\lambda_k'' = \mathrm{SecInv}\left(-e_k''\right). \tag{33}$$

Then, since we cannot know which kind of classification should be biased in the multiclassification condition, $w_{k,i}$ is increased by λ_k times for only the correct classifiers. And for the error classifiers, $w_{k,i}$ stay the same values. S_1 updates the weight vector by computing

$$w_{k+1,i}' = w_{k,i}' \cdot \lambda_k^{1 - I\left(y_i' - p_{k,i}'\right)}. \tag{34}$$

And S_2 computes

$$w''_{k+1,i} = w''_{k,i} \cdot \lambda_k^{I\left(y''_i - p''_{k,i}\right)}. \tag{35}$$

Finally, the output also has a little changes. Set $\xi \in \{\xi_1, \xi_2, \ldots\}$, where ξ is the set of all possible classification results. S_1 and S_2 have to, respectively, compute

$$O(x') = \arg\max_{\xi} \left(\sum_{i=1}^{\infty} \mathrm{ISLog}\left(\mathrm{SecInv}\left(e'_i\right) - 1\right) \cdot I\left(y' - \xi'\right) \right) \tag{36}$$

and

$$O(x'') = \arg\max_{\xi} \left(\sum_{i=1}^{\infty} \mathrm{ISLog}\left(\mathrm{SecInv}\left(e''_i\right)\right) \cdot I\left(y'' - \xi''\right) \right). \tag{37}$$

By multiple invoking the secure comparison function, we can get the classification result with maximum possibility, which is just the result of function $\arg\max_{\xi}(\cdot)$.

VI. Theoretical Analysis of POR

A. Correctness Analysis of POR

In the training process of POR, we preserve the iterative approximation method to compute the nonlinear functions. Therefore, the details of theoretical correctness analysis are given to prove that the differences of outputs between POR and the original AdaBoost learning model is negligible.

We first have to give Lemma 1 which has been proved correct in our previous work.

Lemma 1: SecAdd, SecMul, SecInv, SecCmp, SecExp, SecLog and the protocols made of the linear combination of them are correct.

Proof: The secure matching exponent protocol SME$(\cdot)$ is based on the single-precision float-point type addition which is a commonly used function in operating system. Its outputs is actually another representation format of secret shares. Consequently, it dose not introduce any errors into our framework. Set the output of the secure exponential function SecExp$(\cdot)$ be $\theta + \epsilon$, where θ is the real exponentiation result and ϵ is the computation error close to zero. We have the output of ISExp$(\cdot) = (\theta + \epsilon) \cdot e^{\beta}$, where β is the integer part of input. And the output can be also transformed into ISExp$(\cdot) = \theta \cdot e^{\beta} + \epsilon \cdot e^{\beta}$. It can be discovered that the error of ISExp$(\cdot)$ is $\epsilon \cdot e^{\beta}$. If the iteration number is kept large enough, the error can always maintain at a very low degree, just like what we have done in the processor. Then, by calling our secure format transformation protocol SME$(\cdot)$, the original logarithm is converted into computing SecLog$(\cdot)$ and one times secure addition. On the grounds of nature of logarithm, the conversion cannot introduce any errors. Above all, the subprotocols in Section IV are proved to be correct. And due to the fact that the interactive protocols of POR are linear combinations of the subprotocols, we can guarantee that POR is correct and its error is also negligible. ■

B. Security Analysis of POR

First, we give the definition of privacy in secure multiparty computation as follows [22].

Definition 1: We say that a protocol π is secure if there exists a probabilistic polynomial-time simulator $\mathcal{S}$ that can generate a view for the adversary $\mathcal{A}$ in the real world and the view is computationally indistinguishable from its real view.

Our proof of POR security is based the following three lemmas.

Lemma 2 [22]: A protocol is perfectly simulatable if all its subprotocols are perfectly simulatable.

Lemma 3 [22]: If a random element r is uniformly distributed on Z_n and independent from any variable $x \in Z_n$, then $r \pm x$ is also uniformly random and independent from x.

Lemma 4: SecAdd, SecMul, SecInv, SecCmp, SecExp, SecLog and the protocols made of the linear combination of them are secure.

Based on the above lemmas, we prove that the simulator can generate a view of POR which is computational distinguishable from the real world view for the adversary $\mathcal{A}$.

Theorem 1 The protocol SME is secure in the semi-honest model.

Proof: The view of SME for S_1 is $\mathrm{View}_1 = \{u_1, m_1, m'_1, m''_1, m'_2, e_1, e_2, e\}$. Since e_1, e_2, and e are exponents known by both of two edge servers, they can be regarded as constant that do not influence security. Then, m_i and its splits can be expressed as the multiplication of a constant and u_1 whose result is two uniformly random values. According to Lemma 3, the result of $m'_1 + m'_2$ is still uniformly random. Consequently, View_1 is simulatable and it is unable to find an effective polynomial algorithm to distinguish View_1 and the simulated view for S_1. In a similar way, it can be prove that View_2 is also simulatable and computational distinguishable. ■

Theorem 2: The protocol ISExp is secure in the semi-honest model.

Proof: For the protocol ISExp, the view of S_1 is $\mathrm{View}_1 = \{u_1, \epsilon_1, v_1, \alpha_1, \beta_1, a, a_1, b_1, a_2, p_1, f_1\}$, where $u_1 = \alpha_1 + \beta_1$. According to the system model we set in Section III, the edge servers possess the ability to generate uniformly randoms. Thus, a_1, b_1, and a_2 are all uniformly randoms. Owing to the fact that the intermediate values v_1, p_1 and output f_1 are results of SecExp and SecMul, they are also all uniformly randoms on the basis of Lemma 4. As a consequence, View_1 is simulatable and computationally indistinguishable from the simulated view. In the same way, the simulator can also generate a polynomial-time indistinguishable view for S_2. ■

Theorem 3: The protocol ISLog is secure in the semi-honest model.

Proof: For the protocol ISLog, the view of S_1 is $\mathrm{View}_1 = \{u_1, \epsilon_1, v_1, m_1, e\}$. From Lemma 4 and the proof of SME, ISExp, we have got that v_1, m_1 are uniformly random values. ϵ_1 and e are trivial constants. Thus, same as the security proof of ISExp, we can prove that View_1 and View_2 are simulatable and computationally indistinguishable. ■

Theorem 4: The interactive protocols for the forward stagewise process of POR is secure in the semi-honest model.

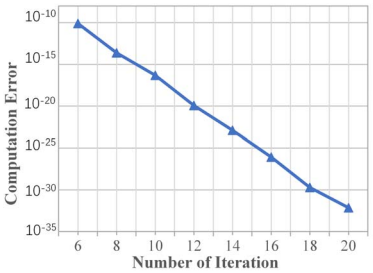

Fig. 4. Computation errors of POR for different iterative numbers.

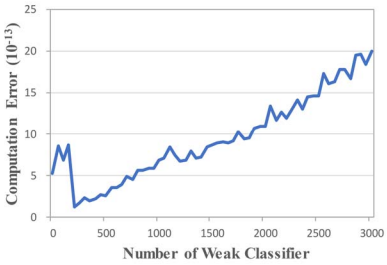

Fig. 5. Computation errors of POR for different ensemble weak classifiers.

Proof: In Section V, we know that there are four interactive subprotocols for the FSW in POR. And the subprotocols can be express as the polynomials based on SecAdd, SecMul, SecInv, SecCmp, SME, ISExp, and ISLog, all of which have been proved to be perfectly similatable. Consequently, according to Lemma 2, we can recursively deduce that Theorem 4 does hold.

Theorem 5: The interactive protocol for the linear addition of weak classifier is secure in the semi-honest model.

Proof: The interactive protocol for the linear addition of weak classifier is only based on SecAdd, SecMul, and SecCmp. Similar to the proof of Theorem 4, we can deduce that Theorem 5 can also hold.

VII. Experiments

In this section, we use experiments to prove the correctness and efficiency of POR and its component protocol, ISExp(·) and ISLog(·). To experiment the performance of POR for face recognition, we adopt the data from the standard face recognition evaluation database FERET [23]. For comparison with the existing DP-based framework, we then choose the same database and weak classifier with [24] for experiments. The original data is encrypted and sent to the two edge servers on a laptop with an Intel Core i5-7200 CPU @2.50 GHz and 8.00 GB of RAM. Two computers equipped with an Intel Core i5-7400 CPU @3.00 GHz and 8.00 GB of RAM are deployed as the edge servers.

A. Performance of POR

Before the experiments, we first select 1000 face images from FERET for the training of AdaBoost. Among them, the training set includes 750 images and the remaining is for test. Each type of their features is with respect to an independent threshold-based weak classifier. And every feature of single image is stored in a 32-bits float, because ciphertext of length $\ell = 32$ is secure enough for most applications. Under this condition, the size of weight vector ω is set to 1×750. For efficiency and practicality, we also deploy the NumPy library of Python to concurrently operate the matrix computation. All of the following experiments use the same configuration, except the one for comparison with the DP method.

Since the interactive protocols of POR comprise secure component functions implemented by iterative series, we evaluate the computation errors of POR in different numbers of iteration round. It can be seen in Fig. 4 that the orders of computation error are almost linearly decreased with the rise of iteration times. This is mainly determined by the nature of Maclaurin Series and Newton–Raphson iterative method. The computation error orders of the two methods are both negatively correlated with the iteration times. And they form the basis of our secure approximate functions. Furthermore, we experiment whether the computation errors of POR have obvious rise in the training process. Set the iteration number of secure component functions be 8. From Fig. 5, it can be observed that the trend of computation errors is exactly ascending, but the rate of increase is quite slow. After addition of about 2000 weak classifiers, the computation error only got one order of magnitude bigger, from 10^{-13} to 10^{-12}. The phenomenon indicates that the effect of iterative training process on computation error is dramatically small and totally tolerable for most applications.

TABLE I
Influence of Data Representation Type for Computation Error

Type	single-precision	double-precision
Reachable Accuracy	10^{-45}	10^{-308}

TABLE II
Runtime and Message Size of POR for Training Set and Test Set

Overhead	Runtime(s)	Message Size(MB)
Training Dataset	249.243	93.16
Test Dataset	2.763	0.29

Moreover, the data representation format is just a variant of the one used in modern computers and does not change the real values of inputs. As long as the inputs and the intermediate computation values are in the representable range, the data representation type and SME(·) do not introduce any extra error but influence the reachable accuracy as illustrated in Table I.

To prove the efficiency of POR, we further examine the communication overhead of POR. As illustrated in the Table II, with 100 weak classifiers which also means 100 times iteration, POR only spend several minutes and no more than 100 millibyte communication load to finish the training job. And due to fact that the test process do not have to iterate but only, respectively, compute the test results of each weak classifier, its runtime and communication load are only about 3 s

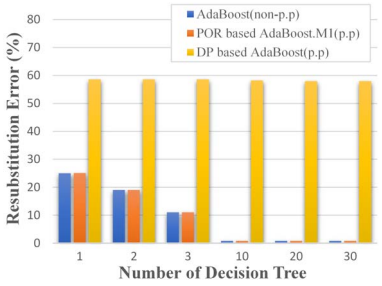

Fig. 6. Performance comparison with DP-based method for training dataset.

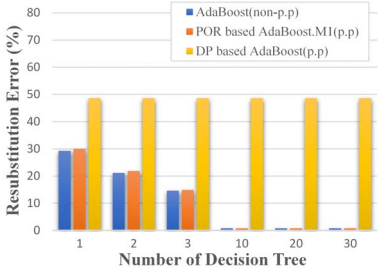

Fig. 7. Performance comparison with DP-based method for test dataset.

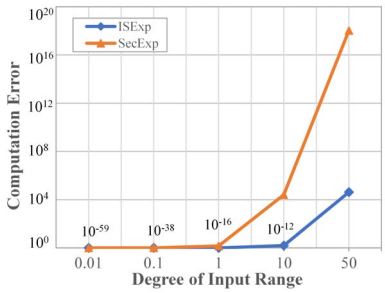

Fig. 8. Computation errors of ISExp and ISLog for different size of input.

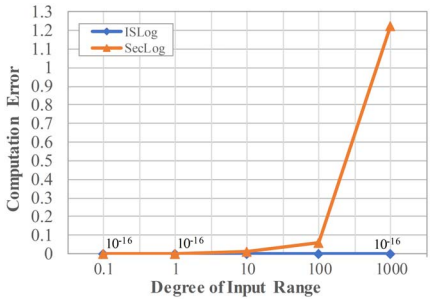

Fig. 9. Computation errors of ISExp and ISLog for different size of input.

and hundreds of kilobytes, which is much less than the training process. Diving to the bottom of the secure functions, we can discover that their overhead are mainly determined by the secure multiplication. Therefore, given the feature data size as $1 \times \mu$ and the overhead of SecMul$(\cdot)$ as τ, we theoretically obtain the time complexity of POR as $\mathcal{O}(\mu \cdot \tau)$, where τ is influenced by the length of data ℓ.

We then compare the computation error of training and testing process with DP method [24] as shown in Figs. 6 and 7. For consistency, the decision tree is chosen as weak classifier in the experiment. And the training data is from a publicly accessible dataset published by the Federal Election Commission for classification of individual economic level [25]. For simplicity, the abbreviation non-p.p in Figs. 6 and 7 presents the original method without privacy preserved and the p.p has the contrary meaning. It can be discovered that POR dramatically outperform the DP-based method under privacy preserved condition. Because of the introduction of Laplace noise, the data utility of DP-based method is obviously affected and the final classification error even reaches almost 60%. Oppositely, our framework only introduce negligible errors into the computation of AdaBoost. Therefore, we can guarantee that the final classification accuracy is the same as the original AdaBoost ensemble result.

B. Performance of Secure Exponentiation and Logarithm Functions

To overcome the input range limit of previous proposed secure functions, we improve the algorithms of the secure exponentiation and secure logarithm functions. Since the two functions are the important components of POR, we evaluate and compare their performance with the previously proposed functions by experiments. Specially, for convenience of comparison, the numbers of iteration round in the following are set to an identical value 20.

Figs. 8 and 9 illustrate the computation error variation of ISExp$(\cdot)$, SecExp$(\cdot)$, ISLog$(\cdot)$, and SecLog$(\cdot)$ with different order of inputs. In Fig. 8, the computation error of ISExp$(\cdot)$ remains negligible until the input closes to 50. Compared to SecExp$(\cdot)$, the upward tendency of computation errors for our method is much slower. And if the inputs are no more than 1, ISExp$(\cdot)$ is able to reach the same precision as SecExp$(\cdot)$. Moreover, when the input is large enough, it can be found that the computation error can always reach an intolerable value. The reason is that, according to the correctness analysis in Section VI, the error grows exponentially and is sure to get the predefined precision at a very quick speed. The only way to solve this problem is to increase the iterative times of the deployed series and delay its appearance. And the phenomenon is also common in the modern processor. Unlike ISExp$(\cdot)$, we do not have worry about the error explosion problem for ISLog$(\cdot)$, because no extra error is introduced by our algorithm compared to SecLog$(\cdot)$. As shown in Fig. 9, the computation error of ISLog$(\cdot)$ is also much lower than SecLog$(\cdot)$, and can always remain the same negligible order.

For evaluation of the efficiency, we examine the runtime and message size of ISExp$(\cdot)$ and SecExp$(\cdot)$ under different length of ciphertext length ℓ. As shown in Figs. 10 and 11, both of the runtime and message size of ISExp$(\cdot)$ are a little larger than SecExp$(\cdot)$. And the differences increase along with the addition of ℓ. This is because two extra secure multiplications are introduced for improvement and the overhead of secure multiplication is strongly influenced by the data length. Meanwhile, we can also see that the extra overhead is rather small and totally acceptable compared to the benefits it provides.

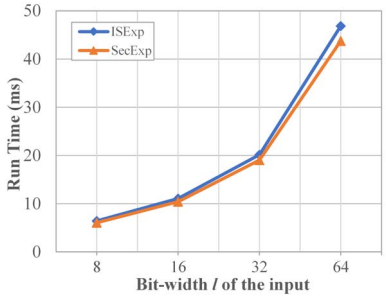

Fig. 10. Runtime of ISExp and SecExp for different length ℓ of the input.

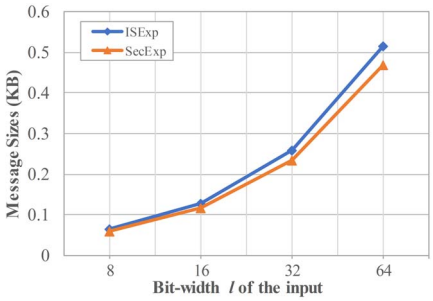

Fig. 11. Communication message sizes of ISExp and SecExp for different length ℓ of the input.

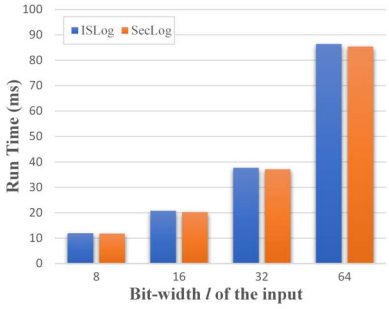

Fig. 12. Runtime of ISLog and SecLog for different length ℓ of the input.

Furthermore, we examine the runtime and message size of ISLog(·) and SecLog(·). To make the comparison clearer, we utilize bar charts to show the experiment results in Figs. 12 and 13. As mentioned before, the extra overhead introduced by ISLog(·) is just some simple local computation which can be finished in no more than one millisecond. Therefore, the differences of overhead between the two functions are almost indistinguishable.

VIII. Related Work

Face recognition has always been a hot-spot problem in the image manipulation field. Especially, in these years, it is increasingly appearing in our daily life. AdaBoost is one of the most outperformed and robust classifier applied in the machine learning-based human face recognition [26]. It can ensemble several weak classifiers to form a stronger one. Ibrahim *et al.* [27] deployed the AdaBoost classifier to recognize the human thermal faces and reach an incredible accuracy 99%. On basis of the nature of AdaBoost, most of its face recognition architectures are hierarchical and

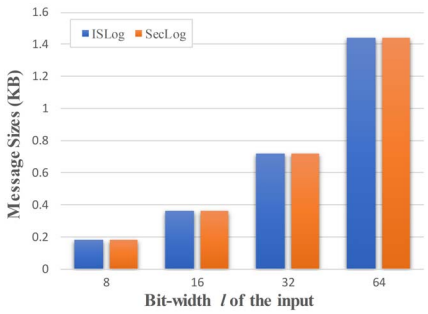

Fig. 13. Communication message sizes of ISLog and SecLog for different length ℓ of the input.

robust [28]. The research of Fan *et al.* [29] illustrated that the principal component analysis (PCA) + AdaBoost method for face recognition is one of the most robust architecture for low resolution images. In addition, there are many other implementation of face recognition, such as the convolutional neural networks [30] and the semi-supervised sparse representation [31] and the trunk-branch ensemble convolutional neural networks (TBE-CNN). And the TBE-CNN also deploy the ensemble method to extract the features of complementary information and patches images of the facial components for the convolutional neural networks.

In the last decade, the privacy of face recognition arrest the attention of researchers and the public with the popularization of automatic face recognition. Until 2014, the privacy-preserving recognition algorithms can be classified into two types: 1) Eigenfaces and 2) SciFi [32]. For Eigenfaces, Erkin *et al.* [33] utilized the multiparty computation to implement a privacy-preserving Eigenfaces recognition on conventional hardware. Sadeghi *et al.* [34] also presented an efficient privacy-preserving face recognition scheme with the HE and Yao's garbled circuits (GC)-based cryptographic building blocks. Later, Schneider and Zohner [35] succeeded to optimize the Goldreich–Micali–Wigderson (GMW) protocols and low the depth of GC to improve the efficient of privacy-preserving face recognition. As for SciFi, in 2010, Osadchy *et al.* [36] first introduced the system SciFi for the privacy-preserving computation of face recognition. It is specific for avoiding the privacy leakage of surveillance in public places. And Bringer *et al.* [37] discovered that the GC-based method outperform the Hamming distance-based SCiFi. In recent years, more advanced techniques are deployed to improve the performance of privacy-preserving face recognition. To protect the privacy of outsourced computation of face recognition in cloud server, Xiang *et al.* [38] proposed an additively homomorphic Paillier encryption scheme. Zhuang *et al.* [39] proposed FRiPAL with dimensionality reduction techniques. Moreover, Mao *et al.* [40] combined deep learning and edge computing for privacy-preserving face recognition. Recently, the DP technology is also deployed to protect the privacy of the image features. Othman and Ross [41] used the DP to disturb the face images for hiding the age, gender and race information. Fan [42] utilized the DP method to add pixelization on images for securely sharing them.

IX. Conclusion

In this paper, we propose an additive secret sharing and edge computing-based lightweight framework for protecting the privacy in face recognition. We first improve the existing secure exponentiation and logarithm functions by getting rid of their limit on input range. Then, we use them to design a series of interactive protocols for privacy-preserving ensemble classification of AdaBoost. These protocols make it possible that all of the image features are computed under encrypted status between the two edge servers. Experiments prove that our framework has more accuracy and higher efficiency than the existing DP-based framework.

References

[1] B. Dosono, J. Hayes, and Y. Wang, "Toward accessible authentication: Learning from people with visual impairments," *IEEE Internet Comput.*, vol. 22, no. 2, pp. 62–70, Mar./Apr. 2018.

[2] W. Mei and W. Deng, "Deep face recognition: A survey," *arXiv preprint arXiv:1804.06655*, 2018.

[3] T. Valentine, *Cognitive and Computational Aspects of Face Recognition: Explorations in Face Space*. London, U.K.: Routledge, 2017.

[4] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "VGGFace2: A dataset for recognising faces across pose and age," in *Proc. 13th IEEE Int. Conf. Autom. Face Gesture Recognit. (FG)*, 2018, pp. 67–74.

[5] P. Li *et al.*, "Privacy-preserving outsourced classification in cloud computing," *Clust. Comput.*, vol. 21, no. 1, pp. 277–286, 2017.

[6] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[7] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[8] Y. Li, Y. Li, K. Xu, Q. Yan, and R. H. Deng, "Empirical study of face authentication systems under OSNFD attacks," *IEEE Trans. Depend. Secure Comput.*, vol. 15, no. 2, pp. 231–245, Mar./Apr. 2018.

[9] K. W. Bowyer, "Face recognition technology: Security versus privacy," *IEEE Technol. Soc. Mag.*, vol. 23, no. 1, pp. 9–19, Jun. 2004.

[10] Y. Wang *et al.*, "Privacy preserving distributed deep learning and its application in credit card fraud detection," in *Proc. 17th IEEE Int. Conf. Trust Security Privacy Comput. Commun. 12th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, New York, NY, USA, 2018, pp. 1070–1078.

[11] T. Wang, Z. Xu, D. Wang, and H. Wang, "Influence of data errors on differential privacy," *Clust. Comput.*, pp. 1–8, Dec. 2017.

[12] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. 38th IEEE Symp. Security Privacy (SP)*, San Jose, CA, USA, 2017, pp. 19–38.

[13] M. Salem, S. Taheri, and J.-S. Yuan, "Utilizing transfer learning and homomorphic encryption in a privacy preserving and secure biometric recognition system," *Computers*, vol. 8, no. 1, p. 3, 2019.

[14] C.-H. Yu, S. S. M. Chow, K.-M. Chung, and F.-H. Liu, "Efficient secure two-party exponentiation," in *Proc. Topics Cryptol. CT-RSA Cryptograph. Track RSA Conf.*, 2011, pp. 17–32.

[15] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Proc. Int. Cryptol. Conf. Adv. Cryptol.*, 1991, pp. 420–432.

[16] I. Damgård, M. Fitzi, E. Kiltz, J. B. Nielsen, and T. Toft, *Unconditionally Secure Constant-Rounds Multi-Party Computation for Equality, Comparison, Bits and Exponentiation* (LNCS 3876). Heidelberg, Germany: Springer, 2006, pp. 285–304.

[17] R. Cramer, I. Damgard, and J. B. Nielsen, "Secure multiparty computation and secret sharing," in *An Information Theoretic Approach*, 2012.

[18] X. Liu, R. Deng, K.-K. R. Choo, and Y. Yang, "Privacy-preserving outsourced support vector machine design for secure drug discovery," *IEEE Trans. Cloud Comput.*, to be published.

[19] IEEE Standards Association, *Standard for Floating-Point Arithmetic*, IEEE Standard 754-2008, 2008.

[20] X. Li *et al.*, "On the soundness and security of privacy-preserving SVM for outsourcing data classification," *IEEE Trans. Depend. Secure Comput.*, vol. 15, no. 5, pp. 906–912, Sep./Oct. 2018.

[21] F. Khodaparast, M. Sheikhalishahi, H. Haghighi, and F. Martinelli, "Privacy preserving random decision tree classification over horizontally and vertically partitioned data," in *Proc. IEEE 16th Int. Conf. Depend. Autonom. Secure Comput. 16th Int. Conf. Pervasive Intell. Comput. 4th Int. Conf. Big Data Intell. Comput. Cyber Sci. Technol. Congr. (DASC/PiCom/DataCom/CyberSciTech)*, 2018, pp. 600–607.

[22] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," in *Proc. Eur. Symp. Res. Comput. Security*, 2008, pp. 192–206.

[23] P. Yang, S. Shan, W. Gao, S. Z. Li, and D. Zhang, "Face recognition using ADA-boosted Gabor features," in *Proc. 6th IEEE Int. Conf. Autom. Face Gesture Recognit.*, 2004, pp. 356–361.

[24] K. Mivule, C. Turner, and S.-Y. Ji, "Towards a differential privacy and utility preserving machine learning classifier," *Procedia Comput. Sci.*, vol. 12, no. 4, pp. 176–181, 2012.

[25] M. Crawley, *Statistics: An Introduction Using R*. Chichester, U.K.: Wiley, 2005.

[26] P.-B. Zhang and Z.-X. Yang, "A novel adaboost framework with robust threshold and structural optimization," *IEEE Trans. Cybern.*, vol. 48, no. 1, pp. 64–76, Jan. 2018.

[27] A. Ibrahim, A. Tharwat, T. Gaber, and A. E. Hassanien, "Optimized superpixel and adaboost classifier for human thermal face recognition," *Signal Image Video Process.*, vol. 12, no. 4, pp. 711–719, 2018.

[28] Z. Zakaria, S. A. Suandi, and J. Mohamad-Saleh, "Hierarchical skin-adaboost-neural network (H-SKANN) for multi-face detection," *Appl. Soft Comput.*, vol. 68, pp. 172–190, Jul. 2018.

[29] S. Fan, Y. Du, J. Zhao, Q. Wang, and F. Guo, "Comparative research of PCA+adaboost and PCA+LDA for face recognition technology," in *Proc. Int. Ind. Informat. Comput. Eng. Conf.*, 2015, pp. 1009–1013.

[30] G. Hu *et al.*, "When face recognition meets with deep learning: An evaluation of convolutional neural networks for face recognition," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, 2015, pp. 142–150.

[31] Y. Gao, J. Ma, and A. L. Yuille, "Semi-supervised sparse representation based classification for face recognition with insufficient labeled samples," *IEEE Trans. Image Process.*, vol. 26, no. 5, pp. 2545–2560, May 2017.

[32] J. Bringer *et al.*, "GSHADE: Faster privacy-preserving distance computation and biometric identification," in *Proc. 2nd ACM Workshop Inf. Hiding Multimedia Security*, 2014, pp. 187–198.

[33] Z. Erkin *et al.*, "Privacy-preserving face recognition," in *Proc. Int. Symp. Privacy Enhancing Technol. Symp.*, 2009, pp. 235–253.

[34] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, "Efficient privacy-preserving face recognition," in *Proc. Int. Conf. Inf. Security Cryptol.*, 2009, pp. 229–244.

[35] T. Schneider and M. Zohner, "GMW vs. YAO? Efficient secure two-party computation with low depth circuits," in *Proc. Int. Conf. Financ. Cryptography Data Security*, 2013, pp. 275–292.

[36] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich, "SCiFI–a system for secure face identification," in *Proc. IEEE Symp. Security Privacy (SP)*, 2010, pp. 239–254.

[37] J. Bringer, H. Chabanne, and A. Patey, "SHADE: Secure hamming distance computation from oblivious transfer," in *Proc. Int. Conf. Financ. Cryptography Data Security*, 2013, pp. 164–176.

[38] C. Xiang, C. Tang, Y. Cai, and Q. Xu, "Privacy-preserving face recognition with outsourced computation," *Soft Comput.*, vol. 20, no. 9, pp. 3735–3744, 2016.

[39] D. Zhuang, S. Wang, and J. M. Chang, "FRiPAL: Face recognition in privacy abstraction layer," in *Proc. IEEE Conf. Depend. Secure Comput.*, Taipei, Taiwan, 2017, pp. 441–448.

[40] Y. Mao *et al.*, "A privacy-preserving deep learning approach for face recognition with edge computing," in *Proc. USENIX Workshop Hot Topics Edge Comput. (HotEdge)*, Boston, MA, USA, 2018.

[41] A. Othman and A. Ross, "Privacy of facial soft biometrics: Suppressing gender but retaining identity," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 682–696.

[42] L. Fan, "Image pixelization with differential privacy," in *Proc. IFIP Conf. Data Appl. Security Privacy*, 2018, pp. 148–162.

Zhuo Ma received the Ph.D. degree in computer architecture from Xidian University, Xi'an, China, in 2010.

He is currently an Associate Professor with the School of Cyber Engineering, Xidian University. His current research interests include cryptography, machine learning in cyber security, and Internet of Things security.

Yang Liu received the B.S. degree in computer science and technology from Xidian University, Xi'an, China, in 2017, where he is currently pursuing the master's degree at the School of Cyber Engineering, Xidian University.

His current research interests include formal analysis of protocols and deep learning neural networks.

Ximeng Liu (S'13–M'16) received the B.Sc. degree in electronic engineering from Xidian University, Xi'an, China, in 2010, and the Ph.D. degrees in cryptography from Xidian University, in 2015.

He is currently a Full Professor with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China. He is also a Research Fellow with the School of Information System, Singapore Management University, Singapore. He has authored or co-authored over 100 research papers include in the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, the IEEE TRANSACTIONS ON SERVICES COMPUTING, and the IEEE TRANSACTIONS ON CLOUD COMPUTING. His current research interests include cloud security, applied cryptography, and big data security.

Jianfeng Ma received the Ph.D. degree from Xidian University, Xi'an, China, in 1995.

He has been a Professor with the Department of Computer Science and Technology, Xidian University, since 1998. He was the Special Engaged Professor of the Yangtze River Scholar in China. His current research interests include cryptology, network security, and data security.

Kui Ren (A'07–M'07–SM'11–F'16) is currently a Distinguished Professor with the School of Computer Science and Technology, Zhejiang University, Hangzhou, China, where he also directs the Institute of Cyber Space Research. His current research interests include cloud and data security, AI and IoT security, and privacy-enhancing technologies.

Mr. Ren is a Distinguished Member of the ACM.