CIT. Journal of Computing and Information Technology, Vol. 33, No. 1, March 2025, 1–24
doi: 10.20532/cit.2025.1005957

A Continuous Authentication Model Based on Improved Flower Pollination Algorithm and Extreme Gradient Boosting

Peng Xiao, Jian Hu, Hailin Wang and Hanruo Li

Information Center Yunnan Power Grid Co., Ltd, Kunming, China

Zero trust systems help to improve the overall security of computer networks, while one of the main challenges of zero trust systems is the construction and optimization of continuous authentication models. Aiming at the shortcomings of the existing authentication performance, this paper proposes a continuous authentication model based on an improved flower pollination algorithm (FPA) and extreme gradient boosting (XGBoost) to improve the accuracy of authentication. The model first uses multiple strategies to optimize FPA to obtain MSFPA; then MSFPA is applied to XGBoost for hyper-parameter tuning to obtain MSFPA-XGBoost; and finally, MSFPA-XGBoost is applied to the user's continuous authentication. Among these approaches, MSFPA incorporates chaotic mapping to enhance the initial population. It also utilizes an adaptive transformation probability strategy to dynamically strike a balance between global and local search. Furthermore, it refines the search equation to optimize both global and local search. Additionally, MSFPA employs a pollen cross-boundary correction strategy and incorporates the concept of cross-mutation to augment the algorithm's exploratory capabilities. The experimental results demonstrate that, in the context of parameter optimization tasks, MSFPA exhibits superior performance compared to other optimization algorithms, specifically in terms of search accuracy and convergence speed. In addition, in terms of continuous identity verification effect, compared with each classification model, MSFPA-XGBoost improves the Accuracy, Recall, Precision, F1-Score, and AUC metrics by an average of 1.84%, 2.49%, 2.33%, 2.39%, and 3.11%, which indicates that the proposed model enhances the accuracy of continuous authentication and is more effectively applicable within zero trust systems.

ACM CCS (2012) Classification: Security and privacy → Security services → Authentication → Multi-factor authentication

Keywords: flower pollination algorithm, extreme gradient boosting, continuous authentication

1. Introduction

Zero Trust [1] represents a novel generation of identity-based security concepts, and its core principle of "never trust, always verify," ensures the security between entities. Comprehensive authentication serves as the foundational prerequisite and cornerstone for the implementation of zero trust systems. Consequently, the precision of user authentication is intimately tied to the overall security of the Zero Trust framework. Systems that use continuous identity [2] for authentication provide a higher level of security with the benefits of non-intrusive, improved user experience, and vulnerability-free against cyberattacks. Currently, many scholars have proposed continuous authentication models. Zhao *et al.* [3] established a recursive neural network based on time and channels to construct an authentication model. Stragapede *et al.* [4] used a triple loss recurrent neural network (RNN) to authenticate the user based on the user's characteristics when performing activities on their mobile phone. Shen *et al.* [5] proposed an authentication model GBDTNN for user touch behavior. The model incorporates a gradient-enhanced decision tree model for handling high-dimensional feature sets and a neural network model for efficient online updating, resulting in a stable online authentication function. While such models possess a certain degree of continuous authentication capability, there remains a need to enhance both the accuracy and performance of the authentication process. Various intelligent algorithms have been applied to the construction process of continuous authentication models, and one of the most effective algorithms is extreme gradient boosting.

Extreme Gradient Boosting (XGBoost) [6] is an iterative decision tree algorithm that uses multiple decision trees with lower classification accuracies to form a combined classifier to improve the overall performance. Liu *et al.* [7] used XGBoost to recognize formation thickness based on log data, and the results showed that the average accuracy of XGBoost in formation thickness and reservoir recognition was up to 95% or more. Xu and Fan [8] proposed a new intrusion detection system model based on logarithmic autoencoder and XGBoost, with detection accuracies of 95.11% and 99.92% on UNSW-NB15 and CICIDS2017, respectively. Yang *et al.* [9] proposed a cross-silo joint XGBoost method to solve the joint anomaly detection problem, which is able to identify anomalies from extremely unbalanced datasets. However, XGBoost is characterized by drawbacks such as an excessive number of parameters and complex computational requirements, highlighting the significance of optimizing its parameters. Among the various algorithms available, the Flower Pollination Algorithm (FPA) stands out as a promising approach for parameter optimization.

FPA, proposed by Yang in 2012 [10], is a novel intelligent optimization algorithm distinguished by its simple structure, minimal parameter requirements, robustness, and adaptability. It possesses the capability to identify optimal solutions. In comparison to other optimization algorithms, FPA exhibits superior speed, stability, and accuracy in complex parameter-tuning tasks. However, in certain specific contexts, FPA is prone to certain limitations, notably an inclination towards premature convergence to local optima and a decrease in the accuracy of identifying the optimal solution. Many scholars have improved it and formed many different versions. Das *et al.* [11] optimize FPA based on dynamic inertia weights of fitness and two popular differential evolution and mutation techniques. Shambour *et al.* [12] used the information about the two solutions randomly selected during the evolutionary process in the global optimization part to tune the detection part of the algorithm to a specific search region. Jia *et al.* [13] proposed a flower pollination optimization algorithm based on cosine cross-generation differential evolution, which used cross-generation differential evolution to guide the local search, and set the cosine inertia weights to improve the search convergence speed. Dao *et al.* [14] integrated FPA and sinusoidal cosine algorithm to overcome the shortcomings of FPA in microgrid operation planning and global optimization. Studies indicate that the proposed HSFPA has improved diversity pollination, and boosted convergence. The aforementioned literature has made certain enhancements to the fundamental FPA. In comparison to the original FPA, a substantial proportion of the modified algorithms demonstrate notable improvements in convergence rates and, to some degree, an elevation in the quality of the resultant solutions. However, the convergence ability and optimization efficiency both have a lot of room for enhancement.

Therefore, to enhance the performance of FPA in optimizing XGBoost parameters and further elevate the performance of XGBoost in continuous authentication, we propose a continuous authentication model that integrates an improved FPA with XGBoost. This model aims to establish a reliable Zero Trust system and ensure the security of computer networks.

The primary contributions of this paper are outlined in greater detail below:

- We introduce an advanced variant of the FPA, designated as Multi-Strategy Flower Pollination Algorithm (MSFPA), which aims to augment the efficacy of FPA in addressing intricate parameter optimization problems. MSFPA mainly optimizes six aspects of FPA: adopting chaotic mapping to improve the initial population; employing a dynamic adaptive strategy for switch probability to maintain a balance between global and local pollination processes; incorporating an adaptive step-size approach to enhance the speed of convergence and precision of the global search optimization; proposing a local search improvement strategy to make full use of the high-quality pollen; adopting the pollen cross-boundary correction strategy to further strengthen the exploration ability of the algorithm; and optimizing the situation of falling into local optimums based on the idea of cross-mutation.

- We propose a continuous authentication model based on MSFPA and XGBoost (called MSFPA-XGBoost). We use MSFPA to tune the hyperparameters of XGBoost, and then train the model according to the optimal parameters and apply it to the continuous authentication of users. The proposed model effectively integrates the strengths of the MSFPA and XGBoost, enabling swift and precise tuning of the intricate algorithm parameters. As a result, it markedly improves the efficacy, precision, and dependability of the ongoing authentication procedure.
- We conduct an experimental comparison of the proposed method with diverse methodologies to further corroborate its feasibility. Specifically, we evaluate the performance of MSFPA against various heuristic optimization algorithms, focusing on its convergence accuracy and speed. We also compare MSFPA-XGBoost with various classification models to assess the performance of MSFPA-XGBoost on accuracy, recall, AUC, *etc.*

The remainder of the paper is structured as follows. Section 2 introduces the flower pollination algorithm. Section 3 describes the continuous authentication model based on the improved flower pollination algorithm and Extreme Gradient Boosting. Section 4 validates the model experimentally, while Section 5 concludes the paper.

2. Flower Pollination Algorithm

This section provides an introduction to FPA for its subsequent multi-strategy optimization. FPA is a heuristic intelligent algorithm based on the collective search patterns of the population, it simulates the various ways that flowering plants reproduce and pollinate. FPA has the qualities of a few parameters, robustness, strong adaptability, *etc.* [15]. Compared with the genetic algorithm (GA), the efficiency of FPA in converging and optimizing is greatly improved. Moreover, by analyzing the global convergence of FPA [16], it can be proved that FPA is effective in parameter tuning and can well optimize the complex parameters of XGBoost. The convergence of FPA is based on randomness and probabilistic control. By reasonably adjusting the ratio of global search to local search, FPA can avoid falling into a local optimal and gradually approach the global optimal solution. FPA ensures that the solution space is fully explored through the continuous flower pollination process, and the accuracy of the solution is gradually improved through local optimization. The time complexity of FPA is mainly proportional to the number of iterations and the number of flowers. If the number of iterations is set to T and the number of flowers is N, the time complexity of the FPA is roughly O(T·N). The algorithmic process of FPA is divided into the following steps:

Step 1. Initialize FPA parameters. The parameters include the maximum iteration number *Maxgen*, population size *N*, and conversion probability *p*.

Step 2. Initialize population. The initial solution $x_1, x_2, ..., x_n$ is randomly generated based on the given upper and lower bounds, and its corresponding fitness value is calculated.

Step 3. Find the best solution $g*$ and its fitness value $f(g*)$ based on the initial population and its fitness value.

Step 4. Generate a new population and determine whether to use global search or local search based on the value of the switch probability $p \in [0, 1]$. Use the global search strategy if the generated random number $rand \in U[0, 1]$ is less than p, otherwise use the local search strategy. The following is the global search equation:

$$x_i^{t+1} = x_i^t + \gamma \cdot levy\left(x_i^t - g*\right),\ rand < p \quad (1)$$

where, x_i^t represents the specific pollen position after the t-th iteration update of the pollen individual i; γ is the step size control factor; levy is the Levy flight step length, and the equation is as follows:

$$levy \approx \frac{\lambda\Gamma(\lambda)\sin\left(\frac{\pi\lambda}{2}\right)}{\pi}\frac{1}{s^{1+\lambda}}, s \gg s_0 > 0 \quad (2)$$

where, λ is a constant, and in this algorithm, $\lambda = 1.5$ is taken; $\Gamma(\lambda)$ is the standard gamma function; s_0 is the minimum step size; s is the step length and is calculated as follows:

$$s = \frac{\mu}{|v|^{1/\lambda}} \quad (3)$$

where, μ and v are two random numbers that satisfy the standard normal distribution $\mu \sim N(0, \sigma^2)$, $v \sim N(0, 1)$, where the variance σ^2 is given by the following equation:

$$\sigma^2 = \left\{\frac{\Gamma(1+\lambda)}{\lambda\Gamma\left(\frac{1+\lambda}{2}\right)} \cdot \frac{\sin\left(\frac{(\pi\lambda)}{2}\right)}{2^{\frac{\lambda-1}{2}}}\right\}^{\frac{1}{\lambda}} \quad (4)$$

The local search equation is as follows.

$$x_i^{t+1} = x_i^t + \varepsilon\left(x_j^t - x_k^t\right),\ rand \ge p \quad (5)$$

where, x_j^t and x_k^t are any two pollinated individuals in the pollen population different from x_i^t during the t-th iteration; ε is a random number that follows the uniform distribution on [0, 1].

Step 5. Update the optimal solution. The updated new solution will be substituted with the existing optimal solution if the fitness value of the flower after the population update is preferable to the present value; otherwise, no adjustments will be made.

Step 6. Check the stop conditions. Repeat Step 4 and 5 until the maximum number of iterations is reached.

The process of global pollination describes the random spread of flowers through the wind, which is an exploratory process. In the updated formula, x_i^t is the current global optimal solution, $g*$ is the position of the current flower, γ is the parameter that controls the search step. According to the point of view of probability theory, the gap between x_i^t and $g*$ presents a generalized exploration process, with the goal of using this difference to steer the search toward a better region.

Local search represents a detailed search in a small area around the current solution. In this process, the solution update formula mainly depends on the gap between the current solution and its neighbors, as well as the step factor of ε. Local search ensures that the algorithm can steadily converge to the local optimal solution.

FPA combines global and local search, where global flower pollination corresponds to a larger step size, emphasizing large-scale exploration, while local flower pollination corresponds to a smaller step size, emphasizing fine optimization near the current solution. By dynamically adjusting the weights of the two, the search for the global optimal solution and the fine optimization of the local optimal solution can be realized.

3. Continuous Authentication Model Based on Improved Flower Pollination Algorithm and Extreme Gradient Boosting

To improve the accuracy of continuous authentication, this paper proposes a continuous authentication model based on FPA and XGBoost, so that it can be better applied to the zero trust system and thus improve the security of computer networks. The model first optimizes the FPA using multiple strategies, then applies it to XGBoost for hyper-parameter tuning, and finally uses the parameter-optimized XGBoost to authenticate the user's identity.

3.1. Improved Flower Pollination Algorithm Based on Multi-strategy

To address the shortcomings of FPA in certain specific contexts, such as its susceptibility to becoming trapped in local optima and its relatively low optimization accuracy, further improvements in both optimization precision and convergence performance of FPA are sought to achieve the purpose of effectively optimizing the complicated parameters of XGBoost. This paper proposes an improved flower pollination algorithm based on multi-strategy (MSFPA), which is mainly improved in the following six aspects.

3.1.1. Initial Population Improvement Strategy

FPA may result in an uneven distribution of individual flower positions because it uses a random method to initialize positions. This has the drawbacks of having low optimization accuracy and being prone to local extremes. Chaotic mapping is distinguished by its randomness, and sensitivity to initial conditions [17–19], enabling it to traverse all possible states within a specified range in a non-repetitive manner, following its inherent rules. Therefore, this paper uses chaos mapping to initialize the position of individual flowers, so that the initial position of individual flowers is distributed more evenly. Chaotic mapping is a dynamic system with irregularities, sensitive initial conditions, periodicity, and infinite detail. A chaotic system is a deterministic system, *i.e.*, its behavior is determined by a set of definite equations, but it exhibits randomness in the details. Chaos mapping has a wide range of applications, especially in optimization problems, because of its global exploration ability and the ability to generate "random" over a large area. Therefore, this paper uses chaos mapping to initialize the position of individual flowers, so that the initial position of individual flowers is distributed more evenly. The steps to initialize the population via chaos mapping for n flower individuals (d-dimensional space) in this paper are as follows.

Step 1. First, randomly generate a d dimensional vector c_i (the first pollen individual) within the [0, 1] interval.

Step 2. Use the logistic map [20] to iteratively generate the remaining $n-1$ vectors. The logistic map equation is as follows:

$$c_{i+1} = \mu c_i \left(1 - c_i\right) \tag{6}$$

where, μ is the control parameter. When $\mu = 4$ is chosen, the logistic map's distribution is at its most uniform, and c_i is the position of the individual flower after chaotic mapping, $i = 1, 2, 3, ..., n-1$.

Step 3. Map the chaotically mapped results to the search area of the solution, using the following equation:

$$x_i = L + c_i \left(U - L\right) \tag{7}$$

where, U and L are the upper and lower boundaries of the search area, and x_i is the initial position of the individual flower in the search space.

Through its ergodic and initial sensitivity, chaotic mapping can cover the entire search space, avoiding the local search blind spots that may be caused by the traditional random initialization method. The position of the flowers can be evenly distributed throughout the search space, so as to ensure that the algorithm has a good global exploration ability at the beginning.

The initial sequence of chaotic mapping is highly random and irregular, which allows the flower pollination algorithm to avoid the local convergence problem caused by the initialization method that only relies on the standard random distribution. Through chaotic mapping, the algorithm can start from a variety of initial points, which enhances the ability to jump out of the local optimum.

3.1.2. Adaptive Switch Probability Strategy

The conversion probability p has a large impact on the performance of FPA. Mahdad and Srairi [21] found that the algorithm has better detection performance when $p = 0.8$. However, maintaining a constant p value during the algorithm's operation hinders the full utilization of the advantages associated with both local and global search capabilities. To address this problem, we integrate the iterations and fitness values to automatically modify the switch probability through Eq. (8), so as to improve the overall optimization-seeking performance of the algorithm. The automatic conversion probability refers to the probability of dynamically adjusting global and local flower pollination based on the mass of the current iteration or the current solution. Generally speaking, the algorithm needs stronger global search ability in the initial search stage and more local optimization ability in the convergence process. The goal of the automatic conversion probability is to adaptively adjust the global and local search to at different stages in order to obtain the optimal solution.

$$p^t = p_{\min} + \left(p_{\max} - p_{\min}\right) \cdot \left(\omega\left(1 - \frac{t}{T}\right) + (1-\omega)\frac{f_{\max}^t - f_i^t}{f_{\max}^t - f_{\min}^t} \right) \tag{8}$$

where the value range of the switch probability p is $p \in [0.2, 0.9]$; $f^t_{\max}$ and $f^t_{\min}$ are the maximum and minimum fitness values in the t-th generation of the population respectively. f_i^t represents the fitness value of the present individual within the population. T is the maximum number of iterations. $p_{\max}$ is the maximum value of parameter p. $p_{\min}$ is the minimum value of parameter p. ω is a weight coefficient used to control the proportion of the two update methods, where $\omega = 0.5$.

In the early stages of the search process, the algorithm is usually in the exploratory phase, where the ability to search globally needs to dominate. The large global pollination probability of p^t allows the flower to jump away from the current solution in the search space in large steps, avoiding falling into the local optimal solution. The automatic conversion probability mechanism can automatically enhance the global search ability in the early stage, so as to improve the global optimization ability of the algorithm.

With the iterative progress of the algorithm, the optimal solution in the search space is gradually discovered, and local search becomes more important. Therefore, gradually decreasing the global pollination probability and increasing the local pollination probability make the position of the flower more finely searched near the current optimal solution, which is helpful to accelerate the convergence and find a more accurate local optimal solution. By gradually decreasing p^t, the automatic conversion probability mechanism increases the proportion of local pollination in the later stage, so that it can focus on a smaller area in the search space and ensure the accuracy of the solution.

3.1.3. Global Search Improvement Strategy

In the global pollination of FPA, the fixed value of the step factor makes the step value lack adaptivity. To address this problem, we use an adaptive step size instead of the original step size in the global pollination part of FPA, and the new step size is defined as follows:

$$levy^t = \begin{cases} levy^{t-1} \times \exp\left(-30 \times \left(T / 2000\right)\right) + 0.0001, \; t > 0 \\ \dfrac{\lambda \Gamma(\lambda) \sin\left(\dfrac{\pi\lambda}{2}\right)}{\pi} \dfrac{1}{s^{1+\lambda}}, \; t = 0 \end{cases} \tag{9}$$

where, T is the number of iterations at their maximum. Early in the algorithm's development, the step size is big, which can help the algorithm better its capacity to find the best solution and hasten its convergence to the area around the ideal flower. As the algorithm progresses, the step size gets smaller, which gives a speedier convergence and higher convergence quality.

The introduction of the adaptive step size mechanism in the flower pollination algorithm can effectively optimize the global pollination process. By dynamically adjusting the step size, the algorithm can better balance the ratio of global search and local search, improve search efficiency, and avoid premature convergence or falling into a local optimal solution. The adaptive step size cannot only accelerate the convergence process, but also enhance the adaptability of the algorithm to different types of optimization problems, making the flower pollination algorithm more flexible and efficient in complex optimization tasks.

3.1.4. Local Search Improvement Strategy

In FPA, local search updates the position by randomly selecting two individuals around the current individual. While this approach facilitates the preservation of population diversity, the identification of the optimal solution may become challenging when individuals exhibit low fitness levels. To solve these problems, this paper proposes an improved local search strategy that redefines the local pollination formula as follows:

$$x_i^{t+1} = x_i^t + \alpha \times \varepsilon \times \left(x_j^t - x_k^t\right) + \left(1-\alpha\right) \times e^{bl} \times \cos\left(2\pi l\right) \times D \times levy^t \tag{10}$$

$$D = \frac{\left|x_{best}^t - x_i^t\right| + \left|x_{secondbest}^t - x_i^t\right| + \left|x_{thirdbest}^t - x_i^t\right|}{3} \tag{11}$$

where, x_i^t is the current pollen individual in the t-th iteration. x_j^t and x_k^t are two solution vectors chosen at random from all the solutions. $x^t{}_{best}$, $x^t{}_{secondbest}$, $x^t{}_{thirdbest}$ respectively represent the top three high-quality pollen individuals in the t-th iteration loop. ε is a random variable of [0, 1]. α is the weight parameter that affects the ratio of the two methods, where $\alpha = 0.5$. b is a constant factor, we take $b = 1$. l is an arbitrary value within [−1, 1].

3.1.5. Pollen Cross-boundary Correction Strategy

In FPA, when a new position is outside the search range, it is usually not processed or replaced with that range, which greatly reduces the optimization efficiency of the algorithm. In this paper, we propose a pollen cross-boundary correction strategy aimed at accelerating the convergence of the algorithm by modifying the parameters of individuals that cross boundaries and directing them toward the globally optimal solution. The new position of the cross-boundary pollen is updated by Eq. (12).

$$\begin{cases} if\ x_i^t\left(r\right) > U, x_i^t\left(r\right) = x_{best}^t\left(r\right) \\ if\ x_i^t\left(r\right) < L, x_i^t\left(r\right) = x_{best}^t\left(r\right) \end{cases} \tag{12}$$

where, $x_i^t(r)$ is the d-th dimension of the i-th operator in the t-th iteration; U and L are respectively the upper and lower boundaries of the search area.

The cross-boundary correction strategy can keep individuals searching in the effective solution space during the global pollination process, and avoid invalid search or search stagnation caused by cross-border. Through the correction of cross-border individuals, the algorithm can explore the solution space more efficiently and increase the probability of finding the global optimal solution.

Without the cross-boundary correction strategy, individuals may cross the search space boundary indefinitely, resulting in an unstable or stagnant search process. The pollen cross-border correction strategy ensures the stability of the search process by limiting the boundaries of individuals. Especially in high-dimensional problems, the cross-border correction strategy can effectively avoid the solution space overflow and ensure that the algorithm continues to search effectively.

3.1.6. Optimization Strategy Based on Cross-mutation

Aiming at the situation that FPA falls into local optimal solutions in the later stage, the idea of cross-mutation [22] is introduced to optimize. In this paper, during the iterative optimization that takes place in the algorithm, if the optimal solution obtained does not change during 5 continuous iterations or the amount of change is less than 0.0001, it can be assumed that the algorithm has entered a local optimal solution. The identified optimal and suboptimal solutions undergo a crossover operation, while the optimal solution undergoes a mutation process to enhance the likelihood of the FPA escaping from local optima.

Cross operation of pollen:

Set pollen $x_1 = (x_{1,1}, x_{1,2}, ..., x_{1,p}, x_{1,p+1}, x_{1,n})$, pollen $x_2 = (x_{2,1}, x_{2,2}, ..., x_{2,p}, x_{2,p+1}, x_{2,n})$, and perform crossoverto:

$$cross(x_1, x_2, p) = \begin{cases} x_{new1} = (x_{new1,1}, x_{new1,2}, \cdots, \\ \qquad x_{new1,p}, x_{new2,p+1}, x_{new2,n}) \\ x_{new2} = (x_{new2,1}, x_{new2,2}, \cdots, \\ \qquad x_{new2,p}, x_{new1,p+1}, x_{new1,n}) \end{cases} \quad (13)$$

where, $x_{i,p}$ represents the p-th dimensional vector of pollen x_i.

The mutation operation of a pollen is:

$$x_{new} = x_{best} + x_{best} \cdot Cauchy(0,1) \quad (14)$$

where x_{new} is the new solution obtained after mutation, x_{best} is the local best value, and *Cauchy*(0, 1) is the standard Cauchy distribution.

The introduction of the idea of cross-mutation can effectively prevent the flower pollination algorithm from falling into the local optimal solution. When individuals converge in a certain region, the crossover generates a new solution by combining the information of multiple solutions to expand the search space. Through the fine-tuning of the solution, the mutation operation tries to sscape the local optimum and avoids search stagnation. The combination of cross-mutations ensures that the search process can maintain the accuracy of local optimization and enhance the global search ability.

3.1.7. Implementation Steps of the Improved Algorithm

Figure 1 displays the flowchart of MSFPA according to the improvement strategies above.

The specific steps of MSFPA are as follows.

Step 1. Initialize parameters. It includes parameters such as maximum iteration number *Maxgen*, dimension *D*, population size *N*, maximum and minimum switch probabilities $p_{\max}$, $p_{\min}$, chaos mapping coefficient μ, changing threshold δ, *etc.*

Step 2. Initialize the pollen population. Use the chaotic map to establish the population's starting location.

Step 3. Evaluate the fitness values of the initial population to determine the optimal pollen position, denoted as g^*, and its associated fitness value, $f(g^*)$.

Step 4. Execute the adaptive switch probability strategy. Calculate the switch probability p using Eq. (8).

Step 5. If $rand < p$, perform global pollination and execute the global search improvement strategy. First, calculate the new step size according to Eq. (9), and then update the position of the pollen individual by Eq. (1); If $rand \geq p$, local pollination is carried out, and the local search improvement strategy is applied to adjust the position of pollen individuals using Eq. (10).

Step 6. Assess whether any individuals within the population exceed their defined boundaries. In the event that they do, implement the pollen cross-boundary adjustment strategy utilizing Eq. (12).

Step 7. Step 7: Evaluate the fitness of the proposed solution. If the fitness of the new solution surpasses that of the previous one, the new solution is selected and its position is updated. Additionally, if the fitness of the new solution exceeds that of the current best overall solution, the best overall solution and its corresponding fitness value are also updated.

Step 8. If the algorithm becomes trapped in a local optimum, the cross-mutation technique is applied to the current best solution to generate a new candidate solution. If this new solution demonstrates superior performance compared to the current optimal solution, it replaces the existing optimal solution; otherwise, no changes are made to the current optimal solution.

Step 9. Verify that the algorithm reaches the end condition, *i.e.*, the maximum number of iterations. If it complies, move on to Step 10; otherwise, go back to Step 4.

Step 10. Output the optimal pollen individual g^* and its corresponding global optimal solution $f(g^*)$, and the algorithm ends.

Algorithm 1 displays the MSFPA pseudocode.

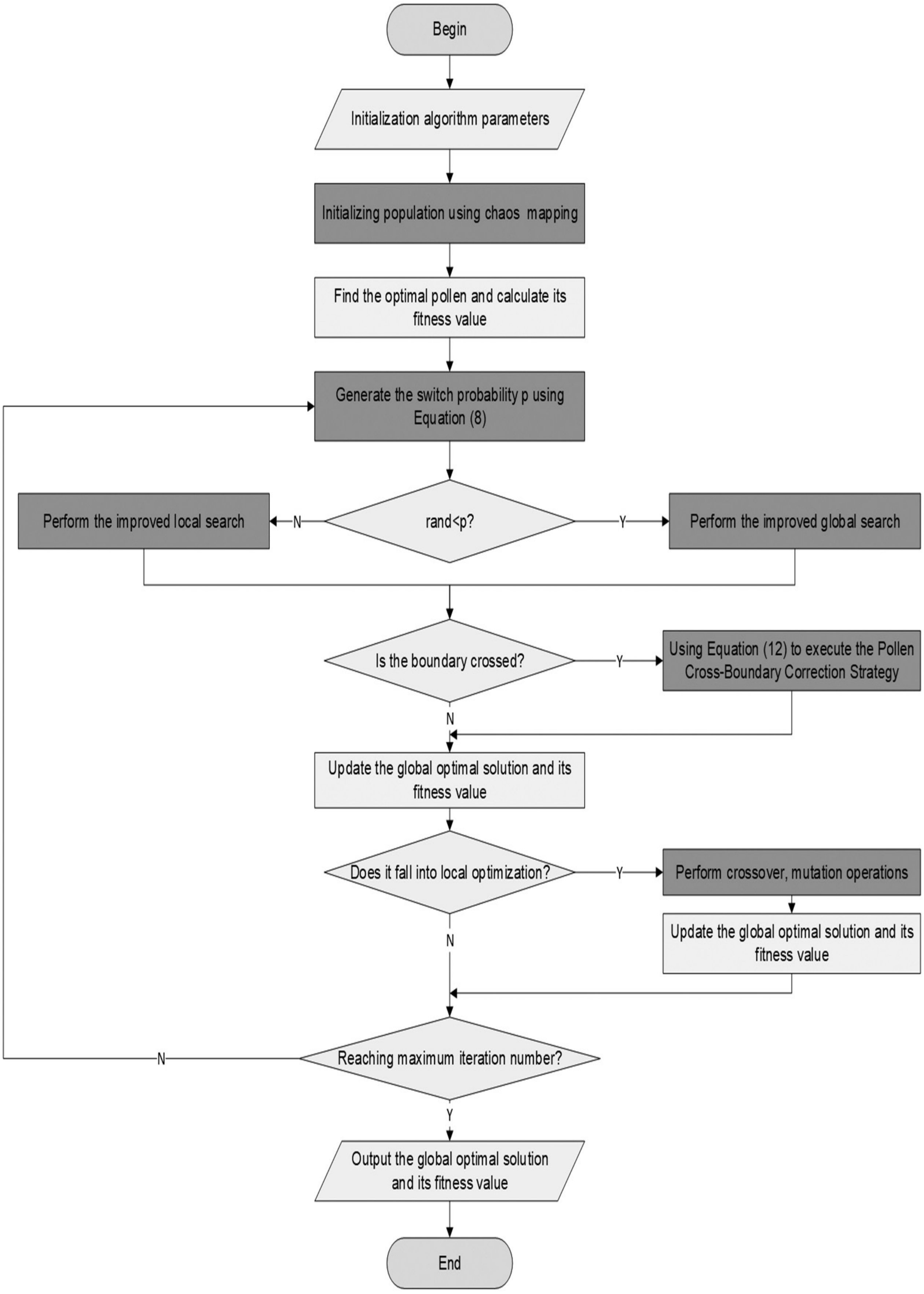

Begin
Initialization algorithm parameters
Initializing population using chaos mapping
Find the optimal pollen and calculate its fitness value
Generate the switch probability p using Equation (8)
rand<p?
N
Perform the improved local search
Y
Perform the improved global search
Is the boundary crossed?
Y
Using Equation (12) to execute the Pollen Cross-Boundary Correction Strategy
N
Update the global optimal solution and its fitness value
Does it fall into local optimization?
Y
Perform crossover, mutation operations
Update the global optimal solution and its fitness value
N
Reaching maximum iteration number?
N
Y
Output the global optimal solution and its fitness value
End

Figure 1. MSFPA flowchart.

Algorithm 1. Pseudocode of MSFPA.

Input: Including population size *N*, maximum iteration times *Maxgen*, dimension *D*, maximum and minimum switch probabilities $p_{\max}$, $p_{\min}$, and upper and lower limits of search space *U*, *L*, chaos mapping coefficient μ, change threshold δ, *etc.*

Output: Optimal pollen individual g^* and optimal fitness value $f(g^*)$.

```
1.  $c_0 = rand(0, 1)$
2.  for $i = 0$ to $N$ do:
3.      $c_{i+1} = \mu c_i (1 - c_i)$  # chaotic mapping
4.      $x_i = L + c_i (U - L)$  # map to the search space of the solution
5.  $[x_{best}, x_{secondbest}, x_{thirdbest}] = sort\_select(X)$  # sort and select the best three pollens
6.  $g^* = x_{best}$
7.  $t = 0, Q = 0$
8.  while $t < Maxgen$ do:
9.      $p^t = p\left( pmin_{max} \cdot \left( \omega\left(1 - \frac{1}{T}\right) + (1-\omega)\frac{f^{t_i^t}_{\max}}{f^{t^t_{\min}}_{\max}()} \right)\right)_{\min}$
10.     $levy^t = adaptive\_step()$  # calculate step
11.     for $i = 0$ to $N$ do:
12.         if $rand(0, 1) < p^t$  # Global Search
13.             $x_i^{t+1} = x_i^t + \gamma \cdot levy(x_i^t - g^*)$
14.         else # Local Search
15.             $\varepsilon = rand(0, 1)$, $l = rand(-1, 1)$, $\omega = 0.5$
16.             $D = \frac{\left|x^t_{best} - x^t_i\right| + \left|x^t_{secondbest} - x^t_i\right| + \left|x^t_{thirdbest} - x^t_i\right|}{3}$
17.             $x_i^{t+1} = x_i^t + \omega \times \varepsilon \times (x_j^t - x_k^t) + (1 - \omega) \times e^{bl} \times cos(2\pi l) \times D \times levy^t$
18.         if $x_i^{t+1} < L$ or $x_i^{t+1} > U$  # Pollen Cross-Boundary Correction
19.             $x_i^{t+1} = boundary\_deal()$
20.         if $f(x_i^{t+1})$ is better than $f(x_i^t)$
21.             $x_i^{t+1} = x_i^{t+1}$
22.         else
23.             $x_i^{t+1} = x_i^t$
24.     $[x_{best}, x_{secondbest}, x_{thirdbest}] = sort\_select(X)$  # sort and select the best three pollens
25.     if $x_{best} == g^*$ or $|x_{best} - g^*| < 0.0001$
26.         $Q = Q + 1$
27.     if $Q == 5$  # judge whether to fall into the local optimization
28.         $x_{tmp} = cross()$  # perform the cross operation
29.         $x_{new} = mutate()$  # perform the mutation operation
30.         if $f(x_{new})$ is better than $f(x_{best})$
31.             $x_{best} = x_{new}$
32.     $g^* = x_{best}$
33.     $t = t + 1$
34. return $g^*, f(g^*)$
```

3.1.8. MSFPA Algorithm Complexity Analysis

Time complexity refers to the measure of computational resources required to execute an algorithm, primarily determined by the frequency of problem repetitions. The time complexity of FPA is mainly affected by the population size *N*, iteration time *Maxgen*, and dimension *D*. According to optimization step of FPA, its time complexity is $O(N \times Maxgen \times D)$. Based on the implementation procedures and pseudocode outlined in Section 3.1.7, the following analysis is conducted regarding the time complexity of the MSFPA: specifically, the time complexity associated with initializing the population through chaotic mapping is $O(N \times D)$, the time complexity of searching for the current optimal pollen is $O(D)$, the time complexity of calculating the adaptive switching probability is $O(D)$, the time complexity of global pollination using global search improvement strategy is $O(D)$, the time complexity of local pollination using local search improvement strategy is $O(D)$. The time complexity of pollen cross-boundary correction is $O(D)$, the time complexity of cross-mutation optimization is $O(D)$. The time complexity of MSFPA in steps 01 to 07 is $O(N \times D)$, and the time complexity of MSFPA in steps 08 to 34 is $O(N \times Maxgen \times D)$, so the total time complexity of MSFPA is $O(N \times Maxgen \times D)$, which is consistent with that of FPA.

The space complexity denotes the amount of storage space necessary for the execution of the algorithm, primarily influenced by the population size *N*, and the dimensionality *D*. Therefore, the space complexity of both MSFPA and FPA is $O(N \times D)$, which means MSFPA does not increase the overhead of the algorithm.

3.2. Optimize XGBoost Parameters Using MSFPA

The accuracy of a model is significantly influenced by the values of its parameters, rendering parameter tuning a crucial phase in model training. However, as illustrated in Table 1, XGBoost encompasses a substantial number of parameters, each serving a distinct purpose. Conventionally, parameter tuning is approached through experience-based, trial-and-error methods [23], which can be both time-consuming and prone to inaccuracies. MSFPA has excellent global optimization ability, fast convergence speed, high precision, *etc.*, which can make up for the shortcomings of XGBoost, such as slow tuning convergence speed of multiple parameters, falling into local optimal solution, large fluctuation of accuracy rate, *etc.* This means MSFPA has the ability to find the optimal XGBoost parameters quickly. The reason for choosing n_estimator is that the value is too small, and the model is easy to underfit and cannot fully learn the rules in the data. This value is too large and can lead to overfitting, where the model performs well on the training set but not well on the test set, and the training time increases significantly. A larger learning_rate can make the model converge quickly, but it may miss the optimal solution, resulting in reduced model accuracy or overfitting. A smaller learning_rate allows the model to learn more carefully, making the model more stable and less prone to overfitting. The larger the maximum depth, the more complex the structure of the tree, the stronger the fitting ability of the model, but the easier it is to overfit and be more sensitive to noise data. The smaller the value, the lower the complexity of the model and the possible underfit. For cases with more features and more complex data, the max_depth can be appropriately increased; for simple data, a smaller max_depth may be sufficient. The min_child_weight value will affect the overfit and underfit of the model, and if there is a lot of noise or outliers in the data, adjusting the value can improve the stability of the model. The value of gamma affects the splitting of the node. The values of subsample and colsample_bytree affect the randomness of the model and how dependent the model is on features.

Hence, this paper introduces an enhanced version of XGBoost, termed MSFPA-XGBoost, which leverages the MSFPA to optimize the hyperparameters of XGBoost, thereby enhancing the model's performance. The flow chart of MSFPA-XGBoost is depicted in Figure 2, which is mainly divided into the following four steps.

Step 1. Input the XGBoost parameter that needs to be optimized.

Step 2. Use MSFPA to optimize each parameter.

Step 3. Examine each parameter to determine whether it is ideal, and if it is, move on to Step 4; otherwise, go back to Step 2.

Step 4. Output the optimized parameter value.

Table 1. XGBoost parameters and defaults.

Parameter	Default Value	Scope	Explain
n_estimator	-	-	The total number of iterations, that is, the number of decision trees.
learning_rate	0.3	[0,1]	Learning rate. It can reduce the weight value of each step, making the model more robust.
max_depth	6	[0,∞]	The maximum depth of the tree. It can be used to prevent overfitting.
min_child_weight	1	[0,∞]	Minimum sum of instance weight (hessian) needed in a child.
gamma	0	[0,1]	Minimum loss reduction is required to further partition a leaf node of the tree.
subsample	1	(0,1]	Subsample ratio of the training instances.
closample_bytree	1	(0,1]	Column sampling rate, that is, feature sampling rate.

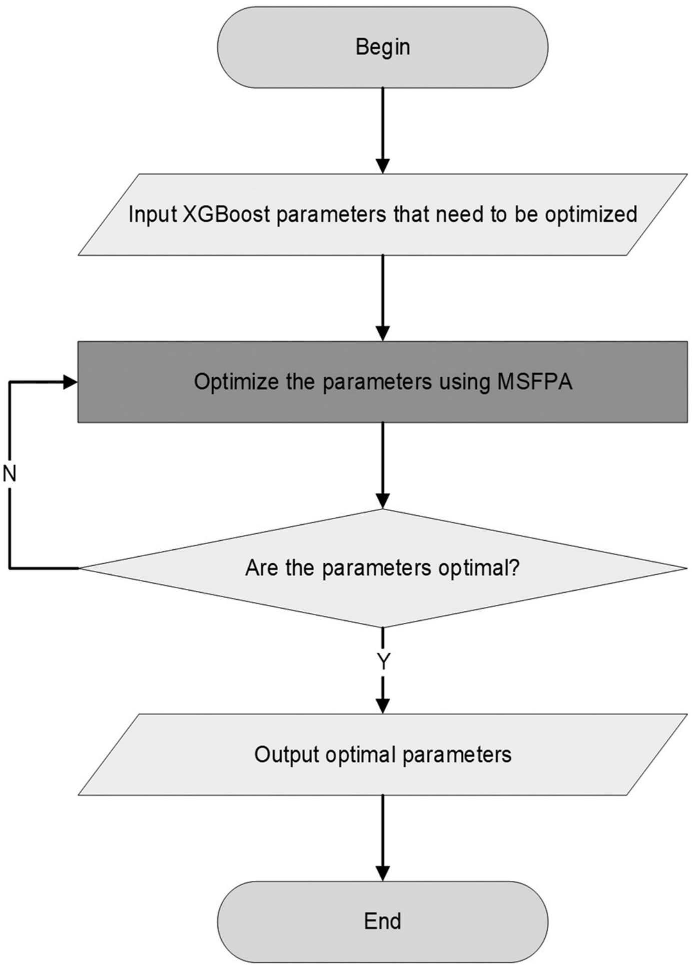

Begin
Input XGBoost parameters that need to be optimized
Optimize the parameters using MSFPA
N
Are the parameters optimal?
Y
Output optimal parameters
End

Figure 2. Flowchart of optimizing XGBoost parameters using MSFPA.

3.3. Continuous Authentication Model

The continuous authentication model based on MSFPA-XGBoost proposed in this paper consists of two parts: the registration stage and the authentication stage, as shown in Figure 3.

3.3.1. The Registration Stage

The registration stage is used to train the user identity models for new users who use the device for the first time, and is mainly divided into four steps:

Step 1. Collect a certain amount of user data using the built-in sensors of the device and perform the preprocessing operations on the data set, such as missing value completion, data deduplication, feature analysis, *etc.*

Step 2. Extract user features based on user data, including gestures, keystrokes, touchscreens, *etc.*

Step 3. Generate MSFPA using multiple strategies to improve FPA.

Step 4. Generate MSFPA-XGBoost using MSFPA Optimize XGBoost parameters.

Step 5. Classify and train user features using MSFPA-XGBoost in order to obtain the user identity model and save it to the database.

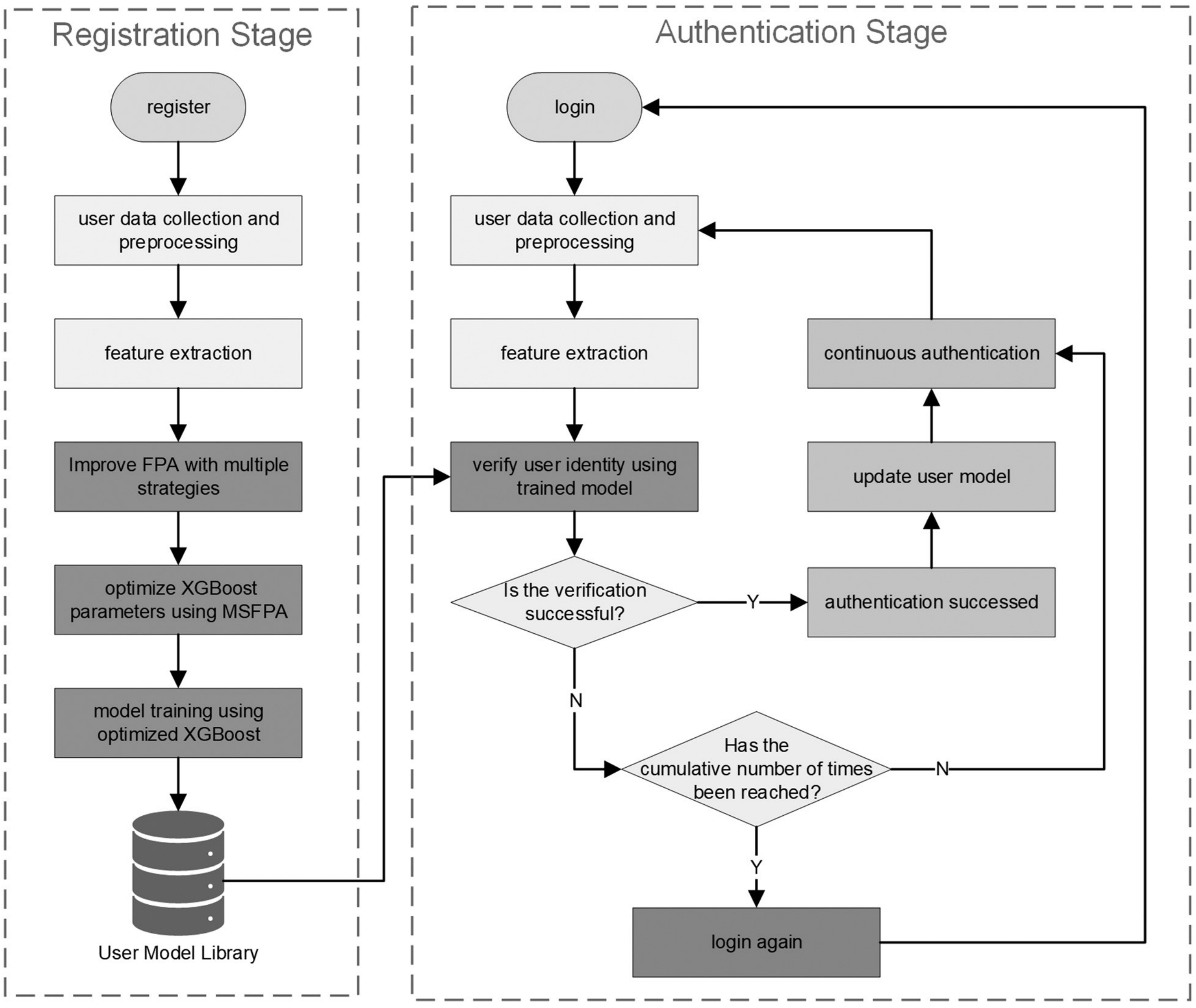

Registration Stage
register
user data collection and preprocessing
feature extraction
Improve FPA with multiple strategies
optimize XGBoost parameters using MSFPA
model training using optimized XGBoost
User Model Library
Authentication Stage
login
user data collection and preprocessing
feature extraction
verify user identity using trained model
Is the verification successful?
Y
authentication successed
update user model
continuous authentication
N
Has the cumulative number of times been reached?
N
Y
login again

Figure 3. Continuous authentication model.

3.3.2. The Authentication Stage

The authentication stage will verify the user's identity continuously and periodically during the user's use of the device, which is mainly divided into five steps:

Step 1. Collect and preprocess user data like the registration stage;

Step 2. Extract user features like the registration stage;

Step 3. Verify the user's identity using the trained model, and perform Step 4 or Step 5 according to the verification results;

Step 4. If the authentication is successful, the current user is regarded as a real user. The user can continue to access the system and go back to Step 1 to proceed to continuous authentication. At the same time, the system saves the current user features, re-classifies training and updates the user identity model after the user exits the system;

Step 5. If the authentication fails, the current user is deemed to be a fake user. When the number of times it is judged to be a fake user is equal to the set cumulative number of times, the current user's permission is locked and the user needs to input the login password to log in to the device again; When the number of times judged to be a fake user is less than the set cumulative number of times, the user returns to Step 1 to restart continuous authentication.

4. Results

4.1. Experimental Environment

The experiments in this paper are conducted on a PC with a Linux operating system, Intel Xeon E5-2680 processor, NVIDIA GeForce RTX3080 graphisc card, and 64GB of memory. The experimental language is Python. The applied experimental environment is Pycharm, and the machine learning frameworks used are Sklearn, XGBoost, *etc.*, which mainly contain Numpy, Pandas, Matplotlib, Scipy and other related libraries.

4.2. Experimental Data

In order to verify the proposed continuous authentication model based on MSFPA-XGBoost in this paper, this paper uses the human activity dataset collected from Yang *et al.* [24] for continuous identity authentication. This dataset collects data recorded by 100 users operating cell phones under different circumstances. The sampling frequency of the data is 100Hz, which mainly includes data such as touch screen, sensors, gestures, *etc.*, as shown in Table 2. The total number of this dataset includes the number of sensor samples 6×10^7, touch samples 4×10^6, gesture samples 1×10^6 and key samples 2×10^5.

4.3. Evaluation Metrics

The evaluation metrics used in the experiments of this paper are as follows:

Accuracy, which indicates the proportion of users who accurately predicted the outcome in terms of the overall number of users.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{15}$$

Recall, which indicates the probability of being predicted as real among all real users.

$$Recall = \frac{TP}{TP + FN} \tag{16}$$

Precision, which indicates the probability that all predicted real users are actually real users.

$$Precision = \frac{TP}{TP + FP} \tag{17}$$

F1 score, which indicates the summed mean of Precision and Recall.

$$F1\text{-}Score = 2 \cdot \frac{P \cdot R}{P + R} \tag{18}$$

Table 2. Dataset details.

Category	Content
Raw touch event	Timestamp, finger count, finger ID, raw touch type, X/Y coordinate, contact size, screen orientation
Tap gesture	Timestamp, tap type, raw touch type, X/Y coordinate, contact size, screen orientation
Scale gesture	Timestamp, pinch type, time delta, X/Y focus, X/Y span, scale factor, screen orientation
Scroll gesture	Starting and ending timestamp, X/Y coordinate, contact size, speed along X/Y-coordinate, screen orientation
Fling gesture	Starting and ending timestamp, X/Y coordinate, contact size, speed along X/Y-coordinate, screen orientation
Key press	Timestamp, press type, key ID, screen orientation
Accelerometer	Timestamp, acceleration force along X/Y/Z-axis

FAR, which indicates the probability that all users that are actually fake are predicted to be real users.

$$FAR = \frac{FP}{FP + TN} \tag{19}$$

The ROC curve and its area AUC enclosed with the X and Y axes in the lower right.

TP denotes true positives, representing the number of instances that are correctly identified as real users; FN stands for false negatives, indicating the number of actual real users that are incorrectly predicted as fake users; and FP represents false positives, which signifies the number of actual fake users that are mistakenly classified as real users. TN represents true negatives, the number of fake users predicted to be fake users.

4.4. Experimental Result

In this paper, two comparative experimental setups are established to underscore the advantages of the proposed methodologies. Specifically, the proposed MSFPA is experimentally benchmarked against various heuristic optimization algorithms, including HSFPA in the study by Dao *et al.* [14], FPA, PSO, GA, *etc.* The proposed MSFPA-XGBoost model is experimentally compared with classification models such as GBDTNN in the study by Zhao *et al.* [3], XGBoost, SVM, *etc.* Comparative experiments are conducted to demonstrate the effectiveness of MSFPA for optimizing XGBoost hyperparameters and the superiority of MSFPA-XGBoost for continuous authentication.

4.4.1. Comparison of MSFPA with Various Heuristic Optimization Algorithms

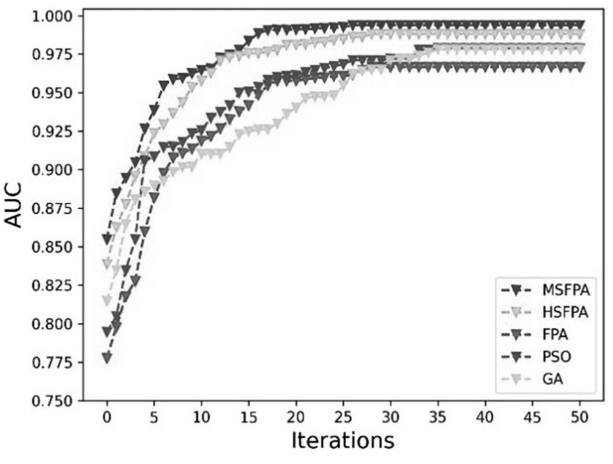
In an attempt to demonstrate the performance superiority of MSFPA, it is experimentally compared with heuristic optimization algorithms such as HSFPA, FPA, PSO, GA, *etc.* In this paper, the four parameters in XGBoost, namely n_estimator, learning_rate, max_depth, and min_child_weight, which have a large impact on the model, are selected as the optimization target parameters, while other XGBoost parameters are default. Given that the algorithm's settings have major effects on its effectiveness, the common parameters of the algorithms should be set to be consistent under the condition of ensuring the same experimental environment and the beginning parameter settings for each algorithm are listed in Table 3. Chaos mapping coefficient adjusts the chaos map strength and balances global and local searches, Maximum switch probability controls the frequency of mode switching, taking into account the diversity and efficiency of searching, Minimum switch probability prevents mode switching from over-frequency and ensures stable searching, Change threshold changes according to the search results, triggering policy adjustment to improve efficiency. Switch probability balances global and local search, and adapts to different search stages, Mutation probability regulates population diversity and avoids premature convergence of the algorithm, Select coefficient controls the selection of excellent individuals, taking into account the speed and diversity of convergence, Step control weight regulates the search step size to suit the search process and accuracy. Learning factor balances the local and global search of particles, avoiding optimization dilemmas, Inertial factor controls the scope and speed of the search, and is adapted to the search phase. Cross probability regulates the generation of new individuals, balancing diversity with the preservation of superior structures. Meanwhile, in order to provide a fair and objective comprehensive evaluation of the convergence ability of each heuristic algorithm, the statistical analysis of the experimental results for each algorithm was conducted using the Wilcoxon signed-rank test [25] (at a significance level of $\alpha = 0.05$) and Friedman's test [26]. Table 4 presents the impact of various algorithms on search accuracy across different evaluation metrics, whereas Figure 4 illustrates the performance of these algorithms in terms of convergence speed.

Table 3. Parameter settings for different algorithms.

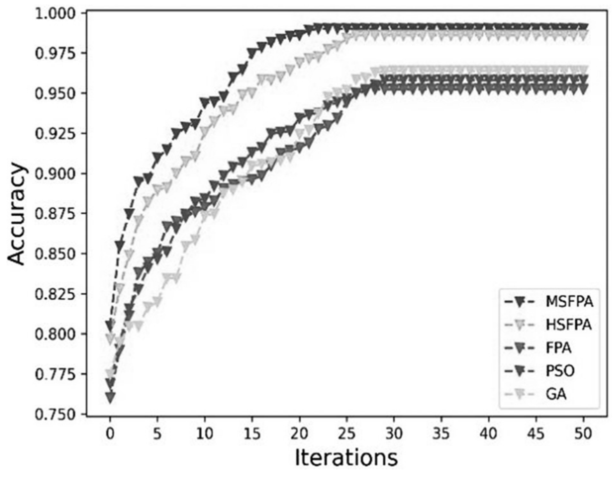
Algorithm	**Parameter**	**Value**
MSFPA	chaos mapping coefficient μ	4
	maximum switch probability $p_{\max}$	0.9
	minimum switch probability $p_{\min}$	0.2
	change threshold δ	0.0001
HSFPA	switch probability p	0.72
	mutation probability q	0.6
	select coefficient ω	0.5
	step control weight $[l_w, h_w]$	[0.21, 0.91]
FPA	switch probability p	0.8
PSO	learning factor c_1	1.5
	learning factor c_2	2
	inertial factor	0.9
GA	cross probability	0.6
	mutation probability	0.01
MSFPA, HSFPA, FPA, PSO, GA	population size N	20
	maximum iterations *Maxgen*	50

Table 4. The iterative results of different algorithms on different metrics.

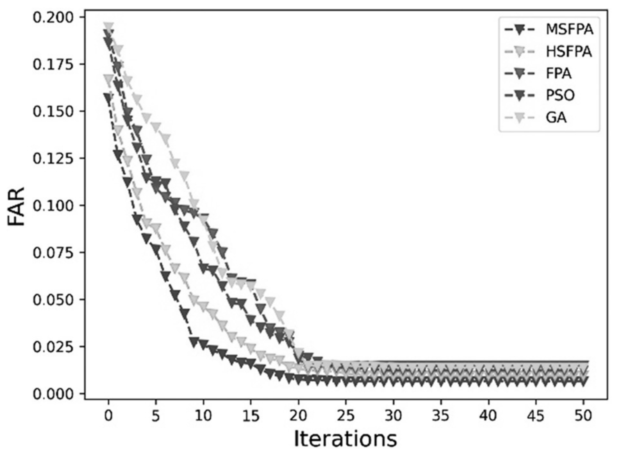
Metric	Algorithm	Max	Mean	Min	Std	Friedman average rank
AUC	MSFPA	0.9937(1)	0.9771(1)	0.8545(1)	0.03173(1)	1
	HSFPA	0.9880(2)	0.9684(2)	0.8386(2)	0.03520(2)	2
	FPA	0.9666(5)	0.9392(5)	0.7775(5)	0.04693(5)	5
	PSO	0.9788(3)	0.9500(3)	0.7945(4)	0.04394(4)	3.5
	GA	0.9780(4)	0.9417(4)	0.8145(3)	0.04099(3)	3.5
Accuracy	MSFPA	0.9900(1)	0.9663(1)	0.8045(1)	0.04176(1)	1
	HSFPA	0.9859(2)	0.9549(2)	0.7745(2.5)	0.04677(2)	2.125
	FPA	0.9524(5)	0.9161(5)	0.7601(5)	0.04777(3)	4.5
	PSO	0.9580(4)	0.9222(3)	0.7687(4)	0.4929(4)	3.75
	GA	0.9638(3)	0.9191(4)	0.7745(2.5)	0.05740(5)	3.625
FAR	MSFPA	0.1565(1)	0.02348(1)	0.006528(1)	0.03452(1)	1
	HSFPA	0.1666(2)	0.03049(2)	0.009528(2)	0.03754(2)	2
	FPA	0.1904(4)	0.04543(4)	0.01422(5)	0.04770(4)	4.25
	PSO	0.1862(3)	0.04101(3)	0.01253(3)	0.04543(3)	3
	GA	0.1944(5)	0.04872(5)	0.01383(4)	0.05342(5)	4.75
Recall	MSFPA	0.9921(1)	0.9676(1)	0.8245(1)	0.04147(1)	1
	HSFPA	0.9899(2)	0.9620(2)	0.8205(2)	0.04407(2)	2
	FPA	0.9620(5)	0.9264(4)	0.8045(3.5)	0.04523(3)	3.875
	PSO	0.9683(3)	0.9269(3)	0.8016(5)	0.04946(4)	3.75
	GA	0.9651(4)	0.9193(5)	0.8045(3.5)	0.04957(5)	4.375
Precision	MSFPA	0.9880(1)	0.9628(1)	0.8045(1)	0.04205(1)	1
	HSFPA	0.9830(2)	0.9551(2)	0.7955(2)	0.04763(2)	2
	FPA	0.9600(4)	0.9187(4)	0.7615(4)	0.05538(4)	4
	PSO	0.9644(3)	0.9280(3)	0.7815(3)	0.04841(3)	3
	GA	0.9590(5)	0.9113(5)	0.7565(5)	0.05922(5)	5
F1-Score	MSFPA	0.9901(1)	0.9652(1)	0.8144(1)	0.04164(1)	1
	HSFPA	0.9865(2)	0.9585(2)	0.8078(2)	0.04572(2)	2
	FPA	0.9610(5)	0.9225(4)	0.7825(4)	0.05036(4)	4.25
	PSO	0.9663(3)	0.9274(3)	0.7914(3)	0.04868(3)	3
	GA	0.9621(4)	0.9152(5)	0.7798(5)	0.05432(5)	4.75



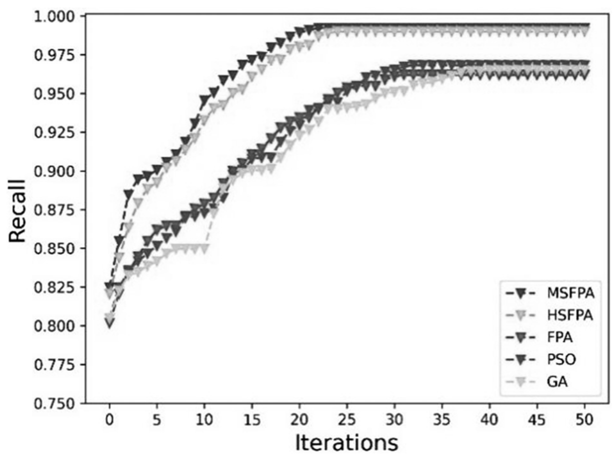
(a)



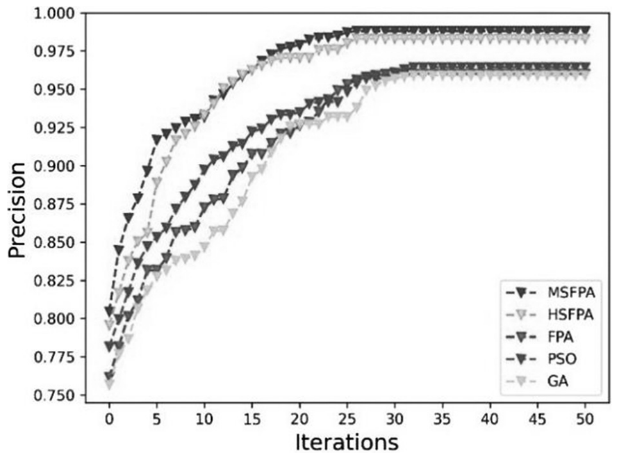
(b)



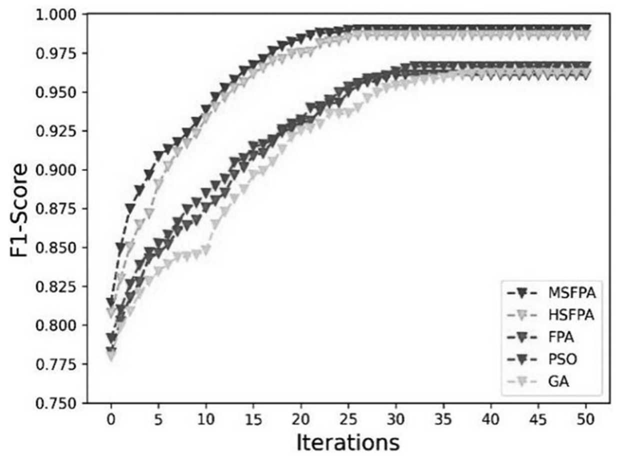
(c)



(d)



(e)



(f)

Figure 4. Convergence of different algorithms on the AUC, accuracy, FAR, recall, precision, F1-Score metrics, as in subfigures: *(a)–(f)*, respectively.

Table 4 demonstrates that the proposed MSFPA has attained favorable outcomes when compared to other algorithms across diverse evaluation metrics. The superiority of MSFPA is evident in its maximum, mean, and minimum values, which suggest improved solution quality, accuracy, and convergence precision. Furthermore, the lower standard deviation associated with MSFPA indicates its more stable performance. Taking the max value as an example, compared to other algorithms, MSFPA improved by 0.58%–2.80% on AUC, 0.42%–3.95% on accuracy, 6.06%–19.50% on FAR, 0.22%–3.13% on recall, 0.51%–3.02% on precision, and 0.36%–3.03% on F1-score. Considering the emphasis of HSFPA on refining local search capabilities, the results obtained after the subsequent iterative refinement process demonstrate a close approximation to those achieved by MSFPA. However, the advancements in MSFPA pertaining to population initialization and global search endow it with superiority over HSFPA in the context of overall optimization. The optimization performance of FPA, PSO, and GA is not much different, but there is still a gap compared to MSFPA and HSFPA. In comparison to the MSFPA, the three other algorithms lag behind by approximately 2.5 percentage points on average in terms of their maximum, mean, and minimum values. Additionally, their larger standard deviations suggest a lesser degree of stability compared to MSFPA. The results of the Friedman test further confirm the superiority of MSFPA, as it received an average rank of 1. Specifically, MSFPA outperformed the HSFPA, FPA, PSO, and GA by an average reduction of 1.02, 3.31, 2.33, and 3.33 ranks, respectively, and the algorithm is ranked first overall. The values in the comprehensive table indicate that the MSFPA in this paper has better search accuracy.

As illustrated in Figure 4, the iteration curve of MSFPA exhibits a downward trend for FAR, while the iteration curves for the remaining indicators display a bias, which indicates that the MSFPA optimization effect is the best. In the latter stages, the iterative curve of HSFPA demonstrates a slight superiority over MSFPA in terms of Precision, primarily due to HSFPA's robust optimization capabilities in local search, but the curve trend of MSFPA outperforms HSFPA in overall convergence and on other evaluation metrics. Although FPA and PSO can converge about 28 iterations, they have fallen into local optima in the later stage. Conversely, while GA possesses the capacity to escape local optima in later stages, it exhibits a slower initial convergence rate. Furthermore, the MSFPA introduced in this study not only exhibits rapid convergence in the early phase, characterized by a steep decline in the curve, but also maintains smoothness in later stages. Notably, it attains optimal convergence after approximately 24 iterations without significant oscillation, demonstrating robust stability. The comprehensive analysis indicates that the MSFPA presented in this paper outperforms in terms of convergence speed.

Table 5 presents the p-values derived from the Wilcoxon test, which were obtained through pairwise comparisons of MSFPA with various algorithms, namely MSFPA versus HSFPA, MSFPA versus FPA, MSFPA versus PSO, and MSFPA versus GA. The null hypothesis assumes no significant difference in performance between the compared algorithms. Alternatively, the alternative hypothesis posits that a difference does exist between the algorithms' values. Notably, all p-values listed in Table 5 are below the 0.05 threshold (indicating a 5% significance level), thereby sufficiently demonstrating a statistically significant difference between MSFPA and the other algorithms. This finding suggests that MSFPA outperforms each of the other algorithms across the evaluated metrics.

In summary, MSFPA can search for optimal parameter groups quickly and accurately, outperforming other heuristic optimization algorithms. The optimal parameter groups after MSFPA tuning obtained by the experiment are shown in Table 6.

As can be seen from Table 6, the best parameter groups obtained on different evaluation metrics are not much different. However, AUC exhibits robustness to variations in the ratio of positive to negative data, making it a suitable metric for evaluating unbalanced datasets compared to other evaluation criteria. Consequently, in this study, the parameter set derived from the AUC evaluation metric is selected as the optimal parameter configuration following the tuning of MSFPA, *i.e.*, n_estimator=325, learning_rate=0.39, max_depth=12, and min_child_weight=1.

Table 5. Comparison of wilcoxon test P-values for MSFPA and other algorithms.

Metric	HSFPA	FPA	PSO	GA
AUC	3.28e-10	3.28e-10	4.42e-10	4.82e-10
Accuracy	2.88e-10	3.28e-10	3.48e-10	3.65e-10
FAR	3.08e-10	3.28e-10	3.47e-10	3.84e-10
Recall	2.46e-10	3.83e-10	4.00e-10	4.90e-10
Precision	3.25e-10	3.47e-10	4.00e-10	4.00e-10
F1-Score	2.88e-10	3.84e-10	4.00e-10	4.90e-10

Table 6. The best parameter population obtained after tuning using MSFPA.

Parameter	AUC (0.9937)	Accuracy (0.9900)	FAR (0.0065)	Recall (0.9921)	Precision (0.9880)	F1-Score (0.9901)
n_estimator	325	322	327	325	328	326
learning_rate	0.39	0.36	0.38	0.37	0.40	0.39
max_depth	12	12	11	11	12	11
min_child_ weight	1	1	2	1	1	2

4.4.2. Comparison of MSFPA-XGBoost with Various Classification Models

In an attempt to reflect the performance superiority of MSFPA-XGBoost, it is experimentally compared with GBDTNN, XGBoost, SVM and other classification models using the same dataset, the data is divided into 80% training set and 20% test set, and the number of training rounds is initially set to 100, which is adjusted according to the convergence of the loss function, the complexity of the dataset, and the computing resources. The parameters of MSFPA-XGBoost use the optimal parameters gained from the experiment in Section 4.4.1, and the other classification models use default values. The performance scores of each classification model are listed in Table 7, and the performance comparison graph is shown in Figure 5, and the ROC curve is shown in Figure 6.

As calculated from Table 7 compared to other models, MSFPA-XGBoost improved by 0.36%-4.82% on accuracy, 0.49%–5.65% on recall, 0.21%–6.07% on precision, 0.09%–5.84% on F1-score, and 0.34%–6.33% on AUC. The average values of improvement on each metric are 1.84%, 2.49%, 2.33%, 2.39%, and 3.11%, respectively. This indicates that MSFPA-XGBoost performs better in terms of continuous authentication compared to other models. At the same time, it can be intuitively seen from Figure 5 that MSFPA-XGBoost outperforms the traditional classification models on all met-

Table 7. Performance Comparison of Different Models.

Model	Accuracy	Recall	Precision	F1-Score	AUC
MSFPA-XGBoost	0.9886	0.9918	0.9871	0.9894	0.9937
GBDTNN	0.9851	0.9870	0.9892	0.9885	0.9903
XGBoost	0.9775	0.9812	0.9768	0.9790	0.9811
Random_Forest	0.9805	0.9762	0.9796	0.9779	0.9667
Decision_Tree	0.9685	0.9568	0.9496	0.9532	0.9483
SVM	0.9431	0.9388	0.9306	0.9348	0.9345

rics. Although GBDTNN is marginally better than MSFPA-XGBoost on Precision, MSFPA-XGBoost outperforms GBDTNN on all other metrics. Also, MSFPA-XGBoost outperforms GBDTNN on AUC is an indication that MSFPA-XGBoost outperforms the unbalanced samples of GBDTNN, proving its superiority in continuous authentication. XGBoost, SVM, Random Forest, and Decision Tree can also achieve more than 90% performance on each metric, but they still have some gap compared to MSFPA-XGBoost, and they lag MSFPA-XGBoost by about 3 percentage points on average. MSFPA-XGBoost outperforms XGBoost on all the metrics also precisely due to the effective optimization of the complex parameters by MSFPA. In addition, as can be seen in Figure 6, the ROC curve of MSFPA-XGBoost is more toward the upper left, further confirming its superiority. In summary, compared to other models, MSFPA-XGBoost has better prediction accuracy, stability and robustness in continuous authentication.

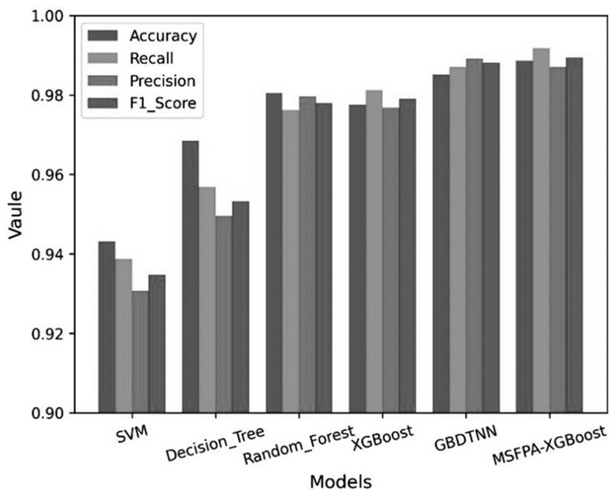


Figure 5. Comparison of different models on different evaluation metrics.

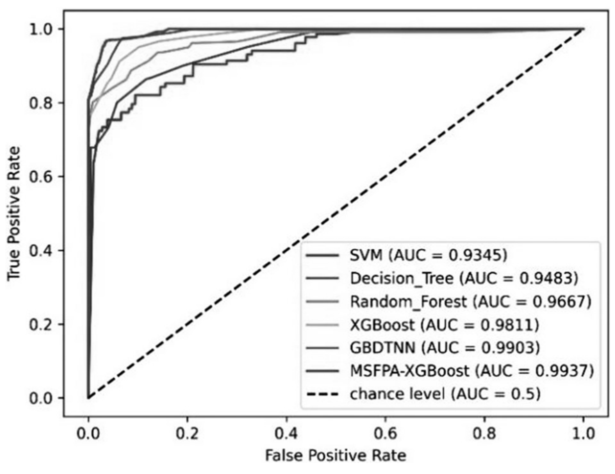


Figure 6. ROC curves of different models.

5. Conclusion

The performance of the continuous authentication model is relevant to the security of a zero trust system, and thus to the security of a computer network. While XGBoost has superior accuracy and speed in continuous authentication scenarios, it has a large number of parameters, and different parameter values have a huge impact on prediction accuracy. The FPA boasts remarkable advantages, including superior global optimization capability, swift convergence rates, and high precision, which facilitate the rapid and accurate tuning of intricate parameters. However, it is prone to falling into local optima and may exhibit reduced optimization accuracy in certain specific scenarios, thereby presenting certain disadvantages. Therefore, this paper proposes a continuous authentication model based on MSFPA-XGBoost, which aims to improve the accuracy of authentication. The proposed model employs a multifaceted approach to address the limitations of each component within FPA. Specifically, it optimizes the initial population, enhances global search capabilities, refines local search strategies, and mitigates the tendency to fall into local optima. Subsequently, the optimized MSFPA is applied to XGBoost for hyper-parameter tuning, and finally, the resulting MSFPA-XGBoost is used for continuous identity authentication of the user's identity. The experimental findings indicate that MSFPA exhibits superior optimization capabilities, precision, and convergence rates when compared to other optimization algorithms, including HSFPA, FPA, PSO, and GA. Furthermore, it demonstrates effectiveness in optimizing parameters such as the number of base classifiers, the learning rate, the maximal tree depth, and the minimum leaf weights of XGBoost on AUC, accuracy, FAR, and other metrics. In addition, the MSFPA-XGBoost has an average improvement of 1.84%, 2.49%, 2.33%, 2.39%, and 3.11% on accuracy, recall, precision, F1-score, and AUC metrics compared to the classification models such as GBDTNN, XGBoost, SVM, *etc.* Additionally, the ROC curve of the model demonstrates a closer proximity to the upper left corner, signifying enhanced accuracy and stability of the model. Therefore, it shows that MSFPA-XGBoost has a good continuous authentication effect, which has good practical application value.

In future work, we will continue to optimize MSFPA and MSFPA-XGBoost and fully incorporate contextual information for continuous authentication so that the model can cope with various complex environments. At the same time, we will supplement the simulated attack experiments to ensure that the security of the system can be improved after being applied to systems such as zero trust.

Declaration of Competing Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was funded by the Innovation Project of China Southern Power Grid Co., Ltd [grant numbers YNKJXM20210198, Research and Application of Key Technologies for Cloud-Edge Collaborative Security in New Power Systems Based on Zero Trust Architecture (Subject 4)].

Data Availability

The dataset used in this paper can be found here: https://wm1693.box.com/s/jkjodaw2scunua7b7qaxt3ker5w9lwt7
(accessed on 6 February 2023).

References

[1] E. Bertino, "Zero Trust Architecture: Does It Help?", *IEEE Secur. Priv.*, vol. 19, no. 5, pp. 95–96, 2021. https://doi.org/10.1109/MSEC.2021.3091195

[2] J. Liu *et al.*, "Risk-based Dynamic Identity Authentication Method Based on the UCON Model", *Secur. Commun. Netw.*, vol. 2022, no. 1, 2022. https://doi.org/10.1155/2022/2509267

[3] C. Zhao *et al.*, "AttAuth: An Implicit Authentication Framework for Smartphone Users Using Multi-modality Data", *IEEE Internet Things*, vol. 11, no. 4, pp. 6928–6942, 2023. http://dx.doi.org/10.1109/JIOT.2023.3314717

[4] G. Stragapede *et al.*, "Mobile Behavioral Biometrics for Passive Authentication", *Pattern Recogn. Lettr.*, vol. 157, pp. 35–41, 2022. https://doi.org/10.1016/j.patrec.2022.03.014

[5] Z. Shen *et al.*, "IncreAuth: Incremental-learning-based Behavioral Biometric Authentication on Smartphones", *IEEE Internet Things*, vol. 11, no. 1, pp. 1589–1603, 2023. http://dx.doi.org/10.1109/JIOT.2023.3289935

[6] T. Chen and C. Guestrin, "Xgboost: A Scalable Tree Boosting System", in *Proc. of the 22nd ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794. https://doi.org/10.1145/2939672.2939785

[7] W. Liu *et al.*, "XGBoost Formation Thickness Identification Based on Logging Data", *Front. Earth Sc-switz.*, vol. 10, 2022. https://doi.org/10.3389/feart.2022.918384

[8] W. Xu and Y. Fan, "Intrusion Detection Systems Based on Logarithmic Autoencoder and XGBoost", *Secur Commun Netw*, vol. 2022, no. 1, 2022. https://doi.org/10.1155/2022/9068724

[9] M. Yang *et al.*, "Achieving Privacy-preserving Cross-silo Anomaly Detection Using Federated XGBoost", *J. Frankin I.*, vol. 360, no. 9, pp. 6194–6210, 2023. https://doi.org/10.1016/j.jfranklin.2023.04.002

[10] X. S. Yang, "Flower Pollination Algorithm for Global Optimization", in *Proc. of the Int. Conference on Unconventional Computing and Natural Computation, Berlin*, 2012, pp. 240–249. https://doi.org/10.1007/978-3-642-32894-7_27

[11] A. Das *et al.*, "Fitness Based Weighted Flower Pollination Algorithm with Mutation Strategies for Image Enhancement", *Multimed. Tools Appl.*, vol. 81, no. 20, pp. 28955–28986, 2022. https://doi.org/10.1007/s11042-022-12879-z

[12] M. D. K. Y. Shambour *et al.*, "Modified Global Flower Pollination Algorithm and Its Application for Optimization Problems", *Interdiscip. Sci.*, vol. 11, pp. 496–507, 2019. https://doi.org/10.1007/s12539-018-0295-2

[13] Y. Jia *et al.*, "A Flower Pollination Optimization Algorithm Based on Cosine Cross-generation Differential Evolution", *Sensors-basel*, vol. 23, no. 2, 2023. https://doi.org/10.3390/s23020606

[14] T. K. Dao *et al.*, "A Hybridized Flower Pollination Algorithm and Its Application on Microgrid Operations Planning", *Appl. Sci.*, vol. 12, no. 13, 2022. https://doi.org/10.3390/app12136487

[15] M. Abdel-Basset and L. A. Shawky, "Flower Pollination Algorithm: A Comprehensive Review", *Artif. Intell. Rev.*, vol. 52, pp. 2533–2557, 2019. https://doi.org/10.1007/s10462-018-9624-4

[16] X. He *et al.*, "Global Convergence Analysis of the Flower Pollination Algorithm: A Discrete-time Markov Chain Approach", *Procedia Com. Sci.*, vol. 108, pp. 1354–1363, 2017. https://doi.org/10.1016/j.procs.2017.05.020

[17] G. Ye *et al.*, "An Effective Framework for Chaotic Image Encryption Based on 3D Logistic Map", *Secur Commun Netw*, vol. 2018, no. 1, 2018. https://doi.org/10.1155/2018/8402578

[18] D. K. Shary and H. J. Nekad, "Position and Speed Control for Permanent Magnet DC Motor Based on Different Optimization Algorithms", *Journal Européen des Systèmes Automatisés*, vol. 57, no. 6, pp. 1705–1711, 2024. https://doi.org/10.18280/jesa.570618

[19] A. Al-Subhi, "Dynamic Economic Load Dispatch Using Linear Programming and Mathematical-based Models", *Mathematical Modelling of Engineering Problems*, vol. 9, no. 3, pp. 606–614, 2022. https://doi.org/10.18280/mmep.090307

[20] H. El Bourakkadi *et al.*, "Enhanced Color Image Encryption Utilizing a Novel Vigenere Method with Pseudorandom Affine Functions", *Acadlore Transactions on AI and Machine Learning*, vol. 3, no. 1, pp. 36–56, 2024. https://doi.org/10.56578/ataiml030104

[21] B. Mahdad and K. Srairi, "Security Constrained Optimal Power Flow Solution Using New Adaptive Partitioning Flower Pollination Algorithm", *Appl. Soft Comput.*, vol. 46, pp. 501–522, 2016. https://doi.org/10.1016/j.asoc.2016.05.027

[22] A. Dockhorn and S. Lucas, "Choosing Representation, Mutation, and Crossover in Genetic Algorithms", *IEEE Comput. Intell. M.*, vol. 17, no. 4, pp. 52–53, 2022. https://doi.org/10.1109/MCI.2022.3199626

[23] Y. Li *et al.*, "Hyper-tune: Towards Efficient Hyper-parameter Tuning at Scale", arXiv preprint arXiv:2201.06834, 2022. https://doi.org/10.48550/arXiv.2201.06834

[24] Q. Yang *et al.*, "A Multimodal Data Set for Evaluating Continuous Authentication Performance in Smartphones", in *Proc. of the 12th ACM Conference on Embedded Network Sensor Systems*, pp. 358–359, 2014. https://doi.org/10.1145/2668332.2668366

[25] H. A. Khan, "ArraySolver: An Algorithm for Colour-coded Graphical Display and Wilcoxon Signed-rank Statistics for Comparing Microarray Gene Expression Data", *Comp. Funct. Genom.*, vol. 5, no. 1, pp. 39–47, 2004. https://doi.org/10.1002/cfg.369

[26] J. Derrac *et al.*, "A Practical Tutorial on the Use of Nonparametric Statistical Tests as a Methodology for Comparing Evolutionary and Swarm Intelligence Algorithms", *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, 2011. https://doi.org/10.1016/j.swevo.2011.02.002

Received: December 2024
Revised: January 2025
Accepted: January 2025

Contact addresses:
Peng Xiao
Information Center Yunnan Power Grid Co., Ltd
Kunming
China
e-mail: xiaopeng03@yn.csg.cn

Jian Hu*
Information Center Yunnan Power Grid Co., Ltd
Kunming
China
e-mail: hujian08@yn.csg.cn
*Corresponding author

Hailin Wang
Information Center Yunnan Power Grid Co., Ltd
Kunming
China
e-mail: wanghailin@yn.csg.cn

Hanruo Li
Information Center Yunnan Power Grid Co., Ltd
Kunming
China
e-mail: lihanruo@yn.csg.cn

PENG XIAO received a bachelor's degree in software engineering from Dianchi College of Yunnan University, Yunnan, China, in 2012, where he is currently the leading technical expert with the Information Center of China Southern Power Grid Yunnan Power Grid Co., Ltd., Kunming, Yunnan, China. His interests focus on information security assessment technology, including network attack and defense technology, network security management, enterprise security system construction, *etc.*

JIAN HU received a master's degree from Yunnan University, Kunming, China, in 2016, where he is currently a technologist with the Information Center of China Southern Power Grid Yunnan Power Grid Co., Ltd., Kunming, Yunnan, China. His research interests focus on information security and machine learning.

HAILIN WANG received an M.S. degree in software engineering from Yunnan University, Yunnan, China, in 2017, where he is currently an engineer in the Information Center of China Southern Power Grid Yunnan Power Grid Co., Ltd., Kunming, Yunnan, China. His interests focus on network security and machine learning.

HANRUO LI obtained a bachelor's degree from Fuzhou University, in 2013.Works in Information Center of China Southern Power Grid Yunnan Power Grid Co., Ltd., engineer. His main research directions include network automatic, net safety construction, *etc.*