International Journal for Housing Science and Its Applications
Publish August 10, 2025. Volume 46, Issue 4 Pages 218-231

https://doi.org/10.70517/ijhsa46417

Artificial Intelligence-Driven Security Risk Identification and Protection Technology for Open Source Software Supply Chain

Qinman Li[1], Xixiang Zhang[1,*], Jing Xie[1], Weiming Liao[1] and Zhezhe Liang[1]
[1] Guangxi Power Grid Co., LTD. Digital Operation Center, Nanning, Guangxi, 530023, China
Corresponding authors: (e-mail: xixixiang@outlook.com).

Abstract Open source software has become an important part of enterprise information systems due to its low cost, openness and transparency, and ease of customization. However, the open source software supply chain faces complex security risks, including management challenges brought about by multi-developer collaboration and difficulties in controlling third-party dependencies, which may lead to data leakage, system paralysis, and business interruption, and bring huge losses to enterprises. This paper proposes an artificial intelligence-driven open source software supply chain security risk identification and protection technology system. The study adopts AHP-entropy combination assignment model to assign weights to supply chain security evaluation indexes, and constructs a security risk identification model based on PSO-SVM, and finally designs a supply chain security protection system based on trusted computing. The results show that the weight of open source code component management is 0.478, which is the most important first-level evaluation index, followed by open source code quality management with a weight of 0.422; among the second-level indexes, open source code submission frequency, self-developed code size and percentage and component vulnerability severity have the highest weights. The PSO algorithm obtains the optimal parameters after 136 iterations, and the constructed risk assessment model has a test set of The assessment accuracy rate reaches 90%, only one sample is misclassified, and the squared correlation coefficient of the regression analysis is 0.96432. The conclusion of the study shows that the combined empowerment method reduces the influence of subjective or objective bias of single empowerment, the PSO-SVM model can accurately identify supply chain security risks, and the end-to-end protection system based on trustworthy computing can realize the trustworthy monitoring of the whole process of business communication, which provides enterprises with a comprehensive and accurate open source software supply chain security management solution.

Index Terms Open source software, supply chain security, risk identification, PSO-SVM, combined empowerment, trusted computing

I. Introduction

With the rapid development of the Internet and the increasing maturity of technology, open source software has become more and more important for enterprises [1], [2]. The advantages of open source software are low cost, openness and transparency, easy customization, etc. Because of this the open source software supply chain exposes a series of security risks [3], [4]. First of all, the development process of open source software generally involves contributions from multiple developers, organizations, and communities, and there is some complexity in the contribution and integration of its source code [5]-[7]. This makes the security of the open source software supply chain more difficult to manage [8]. Second, open source software often relies on other open source libraries or tools, and these third-party dependencies are often difficult to control and ensure their security [9], [10]. Once a third-party dependency is vulnerable or attacked, it will directly affect the security of using open source software [11]. These risks may lead to serious consequences such as data leakage, system paralysis, and business interruption, which will bring huge losses to enterprises [12], [13]. Therefore, enterprises must pay great attention to software supply chain security risks and take effective measures to deal with them.

And with the development and application of artificial intelligence, AI-driven open source software supply chain security risk identification and protection technology has become an important means to deal with new security threats [14], [15]. Artificial intelligence technology has powerful data processing and analysis capabilities, which can help enterprises realize comprehensive monitoring and management of the open source software supply chain and improve the security and reliability of the supply chain [16]-[18]. Artificial intelligence technology can discover abnormal behaviors and potential risks in the supply chain through data mining and analysis, and warn the risks in the supply chain in advance based on intelligent perception and prediction [19]-[21]. In addition, AI technology can

improve the response speed and accuracy of the open source software supply chain and reduce human errors and risks in the supply chain through automation and intelligence [22]-[24].

With the rapid development of the Internet and the increasing maturity of technology, open source software has become more and more important for enterprises. The advantages of open source software are low cost, open and transparent, easy to customize, etc. Because of this the supply chain of open source software exposes a series of security risks. First of all, the development process of open source software generally involves contributions from multiple developers, organizations and communities, and the contribution and integration of its source code also has a certain degree of complexity. This makes the security of the open source software supply chain more difficult to manage. Second, open source software often relies on other open source libraries or tools, and these third-party dependencies are often difficult to control and ensure their security. Once a third-party dependency is vulnerable or attacked, it will directly impact the security of using open source software. These risks may lead to serious consequences such as data leakage, system paralysis, business interruption, etc., which will bring huge losses to the enterprise. Therefore, enterprises must attach great importance to software supply chain security risks and take effective measures to deal with them. And with the development and application of artificial intelligence, AI-driven open source software supply chain security risk identification and protection technology has become an important means to deal with new security threats. Artificial intelligence technology has powerful data processing and analysis capabilities, which can help enterprises realize comprehensive monitoring and management of the open source software supply chain and improve the security and reliability of the supply chain. Artificial intelligence technology can discover abnormal behaviors and potential risks in the supply chain through data mining and analysis, and warn of risks in the supply chain in advance based on intelligent perception and prediction. In addition, AI technology can improve the response speed and accuracy of the open source software supply chain and reduce human errors and risks in the supply chain through automation and intelligence.

This study constructs a complete set of open source software supply chain security risk identification and protection technology system, firstly, it proposes open source software supply chain security evaluation index system, including three first-level indexes and 14 second-level indexes for open source code component management, open source code quality management and open source code source management. Then a combined assignment model combining the hierarchical analysis method and entropy value method is used to assign weights to each indicator, which overcomes the limitations of a single assignment method. Then the support vector mechanism (SVM) risk identification model is improved based on particle swarm optimization algorithm, and the SVM parameters are optimized by PSO algorithm to improve the accuracy and generalization ability of the model. Finally, a supply chain security protection system based on trusted computing technology is designed to realize the entire security protection from the source to the end of the supply chain and the end-to-end trusted construction.

II. Open source software supply chain security risk identification model

II. A. Combinatorial Empowerment Model

II. A. 1) Standardization of data

Normalization, as is often the case, refers to the process of feature scaling in feature engineering. The use of feature scaling can have two effects on data processing: first, features of different magnitudes can be in the same numerical magnitude. Since data with large variance implies a large discrepancy from the mean value of the data, feature scaling can reduce the impact of features with large variance on the system and eliminate model errors as much as possible. The second is to speed up the convergence of the learning algorithm.

The first method is the commonly used z-score normalization process, and the second method is min-max normalization. z-score normalization as the more common normalization method, and therefore also directly called Standardization, the specific method is:

$$x_i^{'} = \frac{x_i - \overline{x}}{\sigma} \tag{1}$$

where $x_i^{'}$ is the normalized value of the variable x, $\overline{x}$ is the mean of the variable x, and σ is the standard deviation of the variable data.

After z-score normalization, the data becomes a distribution with mean 0 and standard deviation 1.

Another more commonly used method, called min-max normalization, often abbreviated as normalization, is:

$$x_i^{'} = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \tag{2}$$

where $x_i^{'}$ is the normalized value of the variable x, $x_{\max}$ is the maximum value of the variable x, and $x_{\min}$ is the minimum value of the variable x. This approach maps the data range into the interval [0, 1]. Also, it is possible to map the data to any interval, for example to map to the interval (a,b) is handled as follows:

$$x_i^{'} = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \tag{3}$$

There is also a less commonly used method called centering, which is done as follows:

$$x_i^{'} = \frac{x_i - \overline{x}}{x_{\max} - x_{\min}} \tag{4}$$

II. A. 2) AHP hierarchical approach

Hierarchical analysis is a decision analysis method that combines quantitative and qualitative analysis proposed by American operations researcher A.L. Satty [25]. Its main steps are as follows:

1) Establish the hierarchical structure model

First of all, according to the objectives of the research problem and the influencing factors of each level, the corresponding hierarchical analysis structure model is constructed.

2) Construct two-by-two judgment matrix

The 1 to 9 scale method proposed by Satty et al. was used to compare different influencing factors at the same level two by two, i.e., the importance of the i factor and the j factor relative to the factors at the previous level was assigned to determine the relative weights of the individual indicator factors, and is expressed as a_{ij}.

3) Hierarchical single sorting and consistency test

Hierarchical single sorting refers to calculating the maximum characteristic root $\lambda_{\max}$ and eigenvector W of each hierarchical judgment matrix one by one, and thereafter obtaining the relative weights of each index respectively. In practical applications, the calculation results of each matrix will have a certain degree of inconsistency. Therefore, in order to improve the scientificity and rationality of the judgment matrix, the following methods are used to test its consistency.

(1) Calculate the consistency indicator CI.

$$CI = \frac{\lambda_{\max} - n}{n-1} \tag{5}$$

If CI=0, it means that the matrix is completely consistent; if CI tends to 0, it means that the matrix has better consistency.

(2) Determine the random consistency index RI to better determine the consistency ratio of the matrix.

(3) Calculate the consistency ratio CR.

$$CR = \frac{CI}{RI} \tag{6}$$

If the CR is less than 0.1, it means that the judgment matrix passes the consistency test, and vice versa, the judgment matrix needs to be adjusted.

(4) Calculate indicator weights

After single-sorting each level, the comprehensive weight of each indicator under the hierarchical analysis method is obtained w_j.

II. A. 3) Entropy method

Entropy value method is a method of analysis under objective conditions, which is now widely used in many fields such as social economy, science and technology and engineering practice [26]. The entropy value method can minimize the subjective influence of the weights of the factors on the evaluation results and make them more in line with the objective reality.

(1) Construct judgment matrix

Based on the information data provided, the judgment matrix of m programs n evaluation indicators is constructed.

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix} \tag{7}$$

(2) Data standardization treatment

The original data were standardized using the extreme value method. Among them, equation (8) is used to deal with positive indicators, and equation (9) is used to deal with negative indicators, and finally get the standardized matrix that has eliminated the differences in the unit of measurement of each indicator.

$$r_{ij} = \frac{x_{ij} - \min(x_{ij})}{\max(x_{ij}) - \min(x_{ij})} \tag{8}$$

$$r_{ij} = \frac{\max(x_{ij}) - x_{ij}}{\max(x_{ij}) - \min(x_{ij})} \tag{9}$$

where x_{ij} represents the j th indicator value of the i th sample $(i = 1, 2, ..., m; j = 1, 2, ...n)$ and r_{ij} is the normalized data. In order to avoid the situation that some of the data after the standardization process have low values or even negative values, thus they are uniformly shifted.

$$r_{ij}^{'} = r_{ij} + H \tag{10}$$

where H is the magnitude of the indicator panning, generally taken as 0.01.

(3) Determine the entropy value of the evaluation indicators

① Normalize the normalization matrix.

$$p_{ij} = \frac{r_{ij}^{'}}{\sum_{i=1}^{m} r_{ij}^{'}} \tag{11}$$

② Calculate the information entropy value e_j for the j th evaluation index.

$$e_j = -\frac{\sum_{1}^{m} p_{ij} ln(p_{ij})}{lnm} \tag{12}$$

③ Calculate the coefficient of variation g_j for indicator j.

$$g_j = 1 - e_j \tag{13}$$

Calculate the weight of the j th indicator as v_j.

$$v_j = \frac{g_j}{\sum_{1}^{n} g_j} \tag{14}$$

II. A. 4) AHP-entropy approach

AHP-entropy value method refers to the use of hierarchical analysis and entropy value method to assign subjective and objective weights to the indicators, and combine the two weights to get more reasonable and objective indicator weights. This method can not only weaken the influence of some subjective factors on the assignment of hierarchical analysis, but also weaken the bias of objective assignment caused by the large differences in the original data information. In order to make the indicator weights more accurate and scientific, the formula (15) is used to calculate the combined weights of the evaluation indicators.

$$w_j^* = \frac{w_j v_j}{\sum w_j v_j} \tag{15}$$

where w_j is the weight of the hierarchical analysis method and v_j is the weight of the entropy method.

II. B.Security risk identification model based on PSO-SVM

II. B. 1) SVM model for risk identification

Let the open source software supply chain risk pattern recognition system consist of m sample data Sample set $X=[X_1,X_2,\cdots X_j,\cdots,X_m]^T$, samples $X_j=[x_{j1},x_{j2},\cdots,x_{ji},\cdots,x_{jn},y_j](j=1,2,\cdots,m)$ consists of n risk metrics observations x_{jm} and 1 risk pattern category y_j with $y_j\in Y$ the pattern category vector, and Y corresponds to the set of rubrics $V=\{v_1,v_2,\cdots,v_l\}(l=1,2,\cdots)$.

Due to the different meanings, scales, and value ranges of the indicator data, data normalization is required. In order to minimize the loss of information during data normalization, the risk model category cut-off point of each indicator is set according to the meaning and value range of the indicator, and the value range of the indicator is thus divided into multiple value intervals to establish a new data normalization method. Let $u_{i(k)}u_{i(k+1)}(k=1,2,\cdots)$ be the vectors corresponding to the risk pattern category cutoffs $x_i(k),x_{i(k+1)}(x_{i(k)}<x_{i(k+1)})$ for the i -th indicator, then The specification of the indicator data x_{ji} is done according to equation (16):

$$x_{ji}^{'}=\begin{cases}u_{i(k)} & x_i\le x_{i(k)}\\ u_{i(k+1)}+(u_{i(k)}-u_{i(k+1)})\dfrac{x_{i(k+1)}-x_i}{x_{i(k+1)}-x_{i(k)}} & x_{i(k)}<x_i<x_{i(k+1)}\\ u_{i(k+1)} & x_{i(k+1)}\le x_i\end{cases} \tag{16}$$

For the normalized sample data $X^{'}=[X_1^{'},X_2^{'},\cdots,X_j^{'},\cdots,X_m^{'}]^T$, where $X_j=[x_{j1}^{'},x_{j2}^{'},\cdots,x_{ji}^{'},\cdots,x_{jn}^{'}]$ is the input vector, and y_i is the output value corresponding to X_j .The SVM passes a nonlinear mapping Φ maps the data X_j to a high-dimensional feature space and performs a linear regression, i.e:

$$f(x)=\omega^T\times\Phi(x)+b \tag{17}$$

where, ω is the weight vector of the hyperplane; b is the bias term.

According to the structural risk minimization principle, Eq. (18) is equivalent to minimizing the cost generalized function:

$$\begin{cases}\min(\dfrac{1}{2}\|\omega\|^2+\dfrac{1}{2}C\sum_{i=1}^{m}\xi_i^2)\\ s.t.\quad y_i-\omega^T\cdot\varphi(x_i^{'})-b=\xi_i\end{cases} \tag{18}$$

where, $\xi\ge 0$, known as the slack variable; $C>0$, the penalty parameter;

$\varphi(x_i^{'})=K(x^{'},x_j^{'})$ is the kernel function, $i,j=1,2,\cdots,m$.

The SVM classification decision function is obtained by transforming to dyadic form and solving using Lagrange operator:

$$f(x)=\sum_{i=1}^{n}a_iK(x^{'},x_j^{'})+b \tag{19}$$

$K(x^{'},x_i^{'})$ is a positive definite function that satisfies Mercer's condition, and the radial basis function (RBF) is the more commonly used kernel function:

$K(x^{'},x_i^{'})=\exp(-\dfrac{\|x^{'}-x_i^{'}\|^2}{2\sigma^2}),\sigma>0$ in the SVM model using RBF. The penalty parameter C and the kernel parameter σ jointly affect the performance of the SVM, so the selection of the optimal SVM parameters has a great impact on the pattern recognition accuracy.PSO has the advantages of fewer parameters, simplicity, and strong global search ability, etc. PSO is introduced to optimize the parameters C and σ .

II. B. 2) PSO algorithm

Let there be a population $Z=(Z_1,Z_2,\cdots,Z_s)$ of s particles in a D -dimensional search space, where the i th particle is denoted as a D -dimensional vector $Z_i=(z_{i1},z_{i2},\cdots,z_{iD})^T$, which represents the particle i in the D -

dimensional vector $Z_i = (z_{i1}, z_{i2}, \cdots, z_{iD})^T$, which represents the position of the particle i in D-dimensional space, i.e., a potential solution of the problem. The fitness value corresponding to each particle position Z_i can be calculated based on the objective function, the velocity of particle i $V_i = (V_{i1}, V_{i2}, \cdots, V_{iD})^T$, and the individual extremum of the population $P_i = (P_{i1}, P_{i2}, \cdots, P_{iD})^T$, and global extremes $P_g = (P_{g1}, P_{g2}, \cdots, P_{gD})^T$, where each particle is updated according to Eq. (20) to update its own velocity and position:

$$\begin{cases} V_{id}^{k+1} = wV_{id}^{k} + c_1 r_1 (P_{id}^{k} - Z_{id}^{k}) + c_2 r_2 (P_{gd}^{k} - Z_{id}^{k}) \\ Z_{id}^{k+1} = Z_{id}^{k} + \alpha V_{id}^{k+1} \end{cases} \tag{20}$$

where, k is the k th iteration; w is the inertia weight; $d = 1, 2, \cdots, D; i = 1, 2, \cdots, s$; c_1, c_2 is the learning factor, a non-negative constant; r_1, r_2 are the random numbers between [0, 1]; α is the the constraint factor that controls the velocity weights.

The inertia weight w reflects the extent to which the particle inherits the previous velocity at the current velocity, and the linear decreasing inertia weight (LDIW) is a more commonly used method:

$$w(k) = w_{start} - (w_{start} - w_{end}) \cdot k / k_{\max} \tag{21}$$

where, w_{start} is the initial inertia weight; w_{end} is the weight for the maximum number of iterations; and $k_{\max}$ is the maximum number of iterations.

II. B. 3) PSO-SVM Recognition Models

The PSO-SVM model construction for open source software supply chain risk pattern recognition is shown in Figure 1.

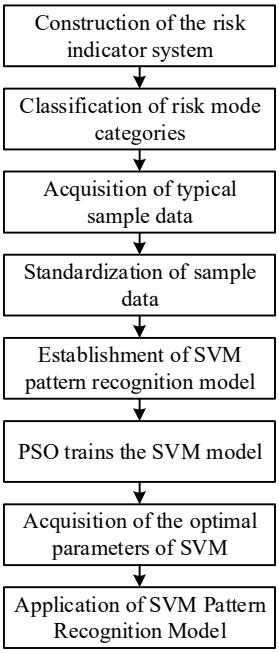

Figure 1: PSO-SVM model for patter recognition

III. Open source software supply chain security risk assessment

III. A. Open source software supply chain security evaluation index system construction

The index system of the open source software supply chain security evaluation model proposed in this paper is adjusted to the first-level indicators: open source code component management, open source code quality management and open source code source management. For the secondary indicators, the design of the model should, on the basis of following the original indicator system of the Chinese standards as much as possible, fully consider factors such as the data accessibility and compliance of the evaluation objects in the actual implementation of the organization, and optimize and adjust the indicators. As the original "Open-source Code source-related Indicators" model was refined from the dimensions of organizational activity, collaborative development, and open-

source community services and support, and indicators such as the original "open-source code richness" were deleted, the final evaluation indicators are shown in Table 1.

Table 1: Safety evaluation index of open source software supply chain

Target	Primary indicator	Secondary indicator
Open source software supply safety index	Open source code component management(A1)	Safe open source component ratio (B1)
		Component vulnerability severity (B2)
		Component license compliance (B3)
	Open source code quality management (A2)	The scale of the self-research code is compared to the proportion of the code (B4)
		Open source code vulnerability rate (B5)
		Open source code vulnerability repair rate (B6)
		The open source code is a serious loophole (B7)
		Open source code pr resolution percentage (B8)
		Open source code bug type issue processing time (B9)
	Source code source management (A3)	Open source software organization activity (B10)
		Open source collaborative development (B11)
		Open source community or project activity (B12)
		Open source community services and support (B13)
		Open source code submission frequency (B14)

III. B. Results and analysis of indicator empowerment

III. B. 1) Determination of subjective weights

The experiment invites a total of five software supply chain security experts to score the importance of each level of indicators, adopts hierarchical analysis to obtain the subjective weight value, takes expert A as an example, constructs the judgment matrix of the level one indicators and calculates it, and the results are shown in Table 2, and finally sums and averages the evaluation results of the weights of the five experts to derive the subjective weight of the final level one indicators, and the results are shown in Table 3. Through the normalization of the optimal parameters, the relative weights of each level of indicators W = (0.291, 0.655, 0.071) are obtained.

Table 2: Primary index judgment matrix

Primary indicator	A1	A2	A3	Weighting	λ max	CR
A1	1	1/4	6	0.291	3.151	0.067
A2	4	1	8	0.655		
A3	1/6	1/8	1	0.071		

Table 3: The primary index is the subjective weight

Experts	Weighting			λ max	CR
	A1	A2	A3		
Z1	0.288	0.648	0.079	3.086	0.067
Z2	0.441	0.491	0.083	3.017	0.014
Z3	0.462	0.459	0.093	3	0
Z4	0.637	0.264	0.113	3.042	0.044
Z5	0.554	0.245	0.215	3.029	0.023
Average	0.476	0.421	0.117	-	

III. B. 2) Determination of objective weights

The data of open source software-related indicators are shown in Table 4.

The data of open source software-related indicators were calculated to obtain the formula for calculating the objective weights of the 14 secondary indicators:

$$\begin{aligned} W &= (w_1, w_2, w_3, \ldots, w_{12}, w_{13}, w_{14}) \\ &= (0.031, 064, 0.053, \ldots, 0.122, 0.025, 0.391) \end{aligned} \tag{22}$$

Table 4: Data on source software

Index		Open source software									
	B1	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
A1	B2	0.898	0.962	1.072	0.998	0.963	0.984	0.97	0.996	0.987	0.994
	B3	0.745	0.489	0.732	0.337	0.178	0.467	0.013	0	0.255	0.253
A2	B4	0.847	0.882	0.799	0.925	0.881	0.979	0.513	0.807	0.968	0.966
	B5	0.995	0.719	0.873	0.95	0.917	0.799	1	0.847	0.998	0.851
	B6	0.017	0	0	0	0.011	0.013	0.391	0	0.006	0.003
	B7	0.436	0.905	1.054	0.807	0.739	0.973	0.85	0.839	0.986	0.837
	B8	0.301	0.605	0.814	0.268	0.358	0.533	0.37	0.258	0.413	0.538
	B9	1	0.998	0.985	0.962	0.941	0.997	0.965	0.996	0.989	0.981
A3	B10	3.013	25.274	26.422	21.128	25.642	38.571	26.415	31.414	2.46	26.792
	B11	73.55	88.2	87.4	97.17	98.1	79.09	78.54	94.03	79.42	91.342
	B12	87.01	89.1	99.03	89.42	88.42	83.42	79.23	87.73	79.83	89.45
	B13	78.14	91.14	87.42	95.1	90.22	83.42	78.63	85.1	84.94	94.12
	B14	80.09	91.28	90.09	89.02	86.154	10.57	81.03	89.52	84.52	88.03

III. B. 3) Determination of portfolio weights

Substituting the objective weights in Table 4 into the formula yields the coefficient of degree of difference R_e =0.498 and the correction coefficient a =0.541, and then the comprehensive weights of the indicators are obtained, as shown in Table 5. From the table, it can be seen that the first-level indicator of open source code component management has the highest weight assigned, with a value of 0.478, indicating that relatively speaking, open source code component management is the most important to experts, and it is also the one that should be emphasized in the identification of security risks in the open source software supply chain. Secondly, the weight of open source code quality management is higher, with a value of 0.422. Among the secondary indicators, the frequency of open source code submission, the scale and proportion of self-study code, and the severity of component vulnerability have the highest weight in the identification of security risks in the open source software supply chain.

Table 5: The primary index is the subjective weight

Primary indicator	Subjective weight	Secondary indicator	Objective weight	Composite weight
A1	0.478	B1	0.019	0.253
		B2	0.052	0.026
		B3	0.043	0.022
A2	0.422	B4	0.056	0.243
		B5	0.007	0.004
		B6	0.043	0.021
		B7	0.036	0.018
		B8	0.013	0.007
		B9	0.038	0.019
A3	0.100	B10	0.041	0.079
		B11	0.023	0.012
		B12	0.112	0.055
		B13	0.019	0.01
		B14	0.498	0.231

III. C. Risk assessment results and analysis

III. C. 1) Matlab Implementation of Supply Chain Risk Assessment Models

In this paper, Matlab 2014a and LIBSVM 3.22 toolbox are selected to write Matlab program to implement PSO-SVM model for virtual supply chain risk assessment. The model parameters are set. The parameters of the support vector machine are set as follows: a Gaussian radial basis kernel function is selected, the kernel parameter $\sigma \in$ (100,1.0), and the penalty parameter $C \in$ (0.1,100).

PSO algorithm parameter settings: particle dimension (σ, C), the maximum number of iterations is 300, the number of populations is 20, take C_1 =1.7, C_2 =1.9, and take the value of fixed inertia weights ω =1. Taking the training results of the training samples and the actual values of the mean square error (MSE) as the particle fitness.

Run the Matlab program of the model, read the sample data of the training set, and calculate the average and optimal adaptation of the particles until the end of the iteration as shown in Fig. 2.

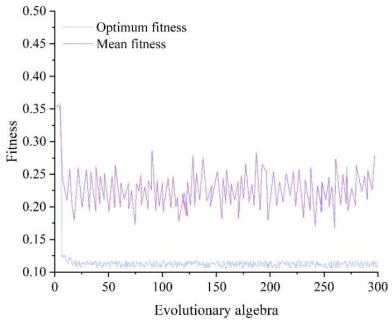

Figure 2: PSO algorithm fitness curve

The optimal parameters after PSO algorithm optimization are C=1.8841, σ = 0.103. The comparison of training results and actual results of the training set is shown in Fig. 3, and the comparison of evaluation results and actual results of the test set is shown in Fig. 4. The mean square deviation of the model assessment results is small, the correlation coefficient is closer to 1, and the model fits well. The risk level value of the test set obtained by using the model assessment is closer to the actual value. As can be seen from the figure, most of the sample assessment results for the test set fall within the actual risk level interval of the corresponding samples, and the accuracy of the assessment of the 10 test set samples is 90%, with one sample being misclassified. The standard PSO-SVM model can accurately assess the risk level of the supply chain during its complete life cycle after its formation based on the current values of the supply chain risk indicators.

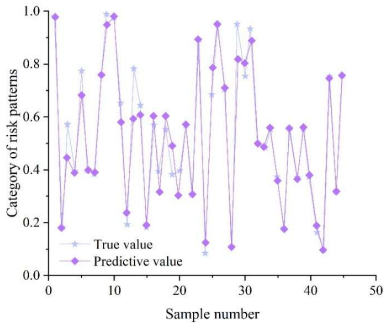

Figure 3: The training set evaluates the comparison of the results

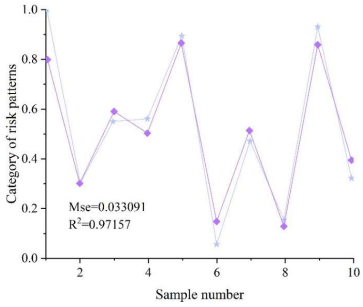

Figure 4: The test set evaluates the comparison of the results

III. C. 2) Accuracy test and error analysis of the evaluation model

The extent of supply chain risk impact and its quantitative values are shown in Table 6.

Table 6: Risk impact of supply chain and its quantification

Risk level	Corresponding risk
Low	[0,0.2]
Lower	(0.2,0.4]
General	(0.4,0.6]
Higher	(0.6,0.8]
Height	(0.8,1]

The output of this model evaluation is: number of iterations iter=136, minimum value obj=-8.74312 obtained from quadratic programming solution converted from SVM file, constant term b of the judgment function is rho=-0.150127, number of support vectors nSV=45, number of support vectors on the boundaries nBSV=14. MSE=0.031201(regression), squared correlation coefficient=0.96432(regression).), squared correlation coefficient=0.96432(regression). The evaluation results are shown in Table 7, and the results show that the assessed risk level of PSO-SVM model is closer to the actual risk level with higher accuracy.

Table 7: Evaluation result

Supply chain number	Actual risk level	The actual area	Assess risk level	Assessment area
Test 1	1.000	Height	0.801	Highet
Test 2	0.311	Lower	0.308	Lower
Test 3	0.561	General	0.412	General
Test 4	0.574	General	0.513	General
Test 5	0.895	Highet	0.893	Highet
Test 6	0.058	Low	0.154	Low
Test 7	0.472	General	0.511	General
Test 8	0.158	Low	0.127	Low
Test 9	0.937	Height	0.893	Height
Test 10	0.312	Lower	0.397	Lower

IV. Trusted computing-based supply chain security protection system

IV. A. Credible Computing

Trustworthy computing is a technology that guarantees the predictability of information systems, which refers to the security protection while calculating, so that the results of the calculation are always the same as the expected value, so that the whole calculation is measurable and controllable, without interference.

The development of trusted computing has gone through several stages. The initial trusted 1.0 comes from computer reliability, mainly by means of troubleshooting and redundant backup, which is a security measure based on fault-tolerant methods. Trusted 2.0 is marked by TPM1.0 introduced by Trusted Computing Group (TCG), which mainly takes hardware chip as the root of trust, and takes trusted metrics, trusted storage, and trusted reports as means to realize single-computer protection of computers. The shortcomings are: it does not consider the security problem from the computer architecture level, and it is difficult to realize active defense [27]. China's trusted computing technology has developed to the 3.0 stage of the “active defense system”, to ensure that the whole process can be measured, controlled and not be interfered with, i.e., parallel defense and computing “active immunity computing mode”. The system architecture is based on password, chip as a pillar, motherboard as a platform, software as the core, network as a link, application into a system to provide security for the application execution environment and network environment. The basic principle of Trusted Computing 3.0 is shown in Fig. 5. Since the platform is powered up, the trust is transferred from TPCM (Trusted Platform Control Module) to the operating system layer by gradually constructing a trust chain to ensure the trustworthiness of the whole information system from the source.

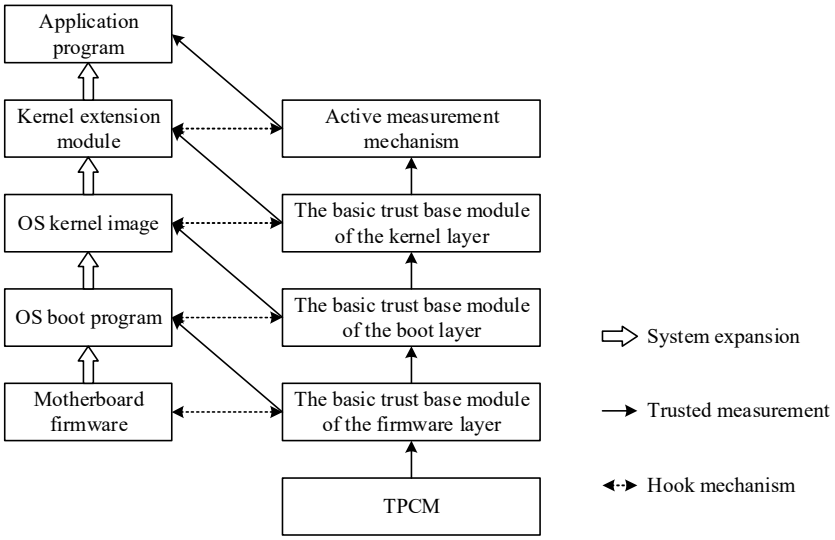

Figure 5: Schematic diagram of the reliable calculation of 3.0

IV. B. Architectural design

IV. B. 1) Trusted supply chain controls

The power system software supply chain control program can learn from the technical route of Apple's application software installation control, applying trusted computing technology and commercial confidentiality system to the power monitoring system, and constructing a full security protection system from the source to the terminal of the supply chain as shown in Figure 6.

Apple requires the developer to sign the software installation package with the certificate issued by the developer during the application software installation process, and the IOS device verifies the signature based on the trusted computing technology before the software installation package is installed and started to ensure that the app has not been tampered with since the installation or the last update. Similarly, in the electric power system, the trust chain transfer mechanism of trusted computing technology can also be used to realize the control of devices and software versions, ensuring that uncertified devices and software versions cannot be executed.

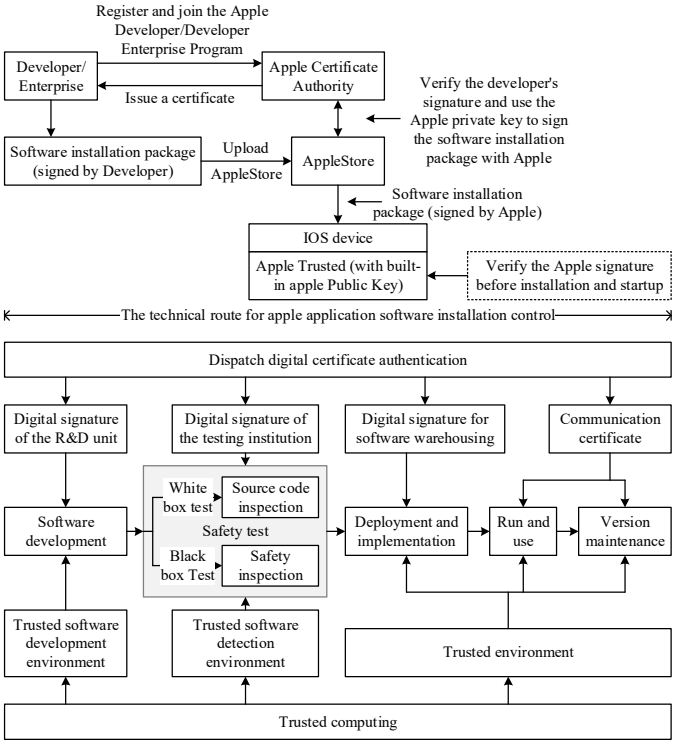

Figure 6: The overall technical framework of trust management of supply chains

IV. B. 2) End-to-end trusted construction

The purpose of end-to-end trusted construction is to ensure that in the grid environment, regardless of the downstream type of business (such as control instructions, remote configuration issuance) or upstream type of business (such as terminal access, data collection and access), the legality and security of the access connection can be realized for the whole process of monitoring and protocol authentication. In Figure 7, by building an end-to-end trusted authentication module and terminal trusted transformation in the trusted management center, the system is able to carry out trusted monitoring and control of the entire business communication process, ensure the integrity, confidentiality and compliance of key business data, and prevent malicious attacks, data tampering and improper access.

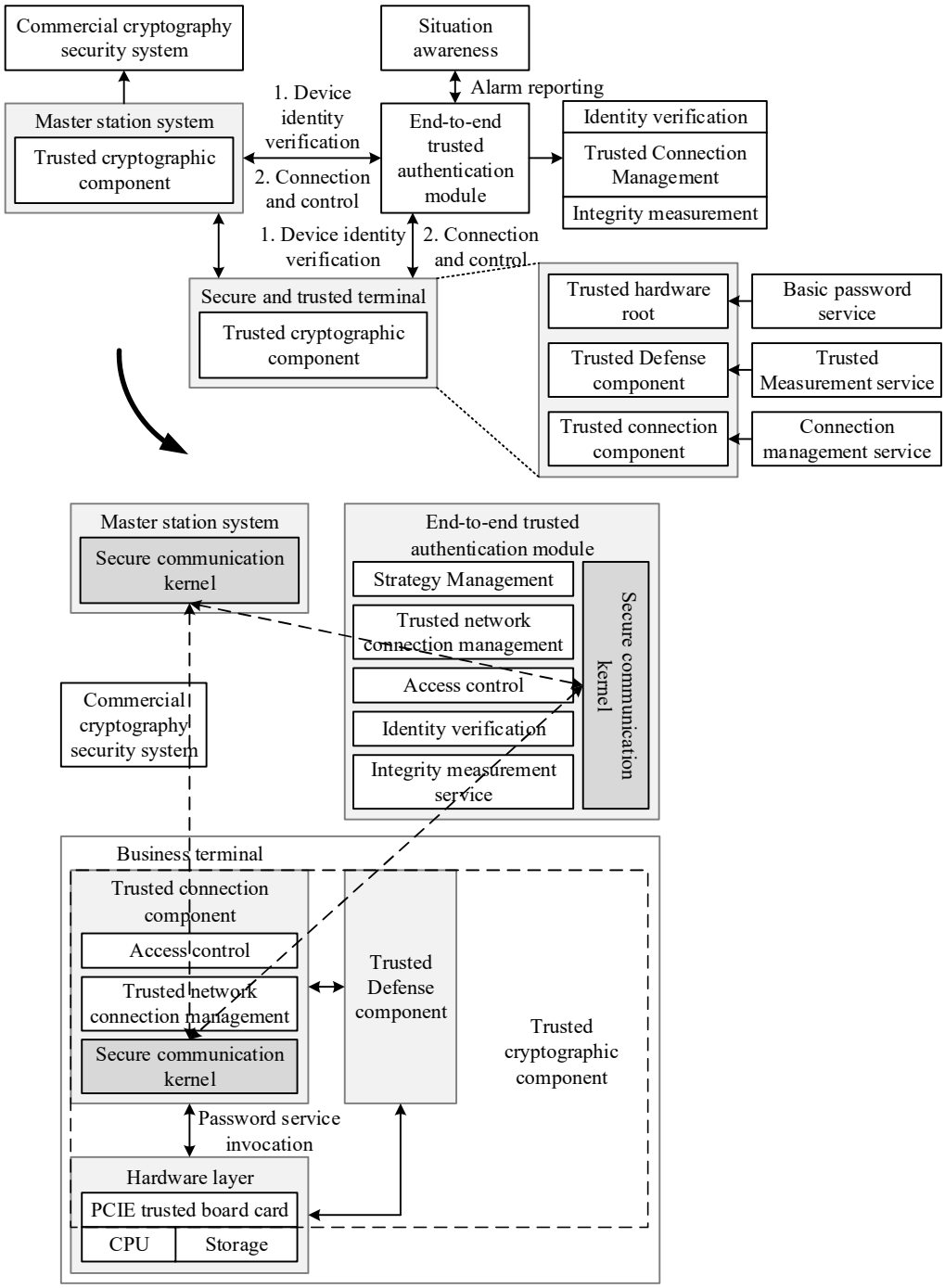

Figure 7: End-to-end credible overall technical plan

V. Conclusion

This study constructs an artificial intelligence-driven open source software supply chain security risk identification and protection technology system, and the main conclusions are as follows:

The weights of open source software supply chain security evaluation indicators are assigned by AHP-entropy combination assignment model, and the results show that open source code component management is the highest-weighted first-level indicator, which takes the value of 0.478, followed by open source code quality management with a weight of 0.422, and open source code source management with a weight of 0.100. Among the second-level indicators, the frequency of open source code submission, the size and percentage of self-developed code and component vulnerability severity rank in the top three, which indicates that enterprises should focus on risk management in these aspects.

The risk identification model constructed based on PSO-SVM shows high accuracy. The model was iterated 136 times by the PSO algorithm to obtain the optimal parameters, and the accuracy of sample evaluation on the test set was 90%, with only one sample being misclassified. The mean square error MSE of the model is 0.031201, and the squared correlation coefficient is 0.96432, which indicates that the model fits well and can accurately assess the risk level of the open source software supply chain.

The supply chain security protection system based on trusted computing technology realizes the whole security protection from the source to the end of the supply chain. The system draws on the technical route of Apple's application software installation control, utilizes the trust chain transfer mechanism of trusted computing technology to realize the control of equipment and software version, and ensures the security of the business communication process through the end-to-end trusted construction, effectively preventing malicious attacks, data tampering and improper access.

This study provides theoretical guidance and practical solutions for enterprise open source software supply chain security management, which is of great significance for improving enterprise information system security.

References

[1] Andersen-Gott, M., Ghinea, G., & Bygstad, B. (2012). Why do commercial companies contribute to open source software?. International journal of information management, 32(2), 106-117.

[2] Shahrivar, S., Elahi, S., Hassanzadeh, A., & Montazer, G. (2018). A business model for commercial open source software: A systematic literature review. Information and Software Technology, 103, 202-214.

[3] Russo, D. (2016, January). Benefits of open source software in defense environments. In Proceedings of 4th International Conference in Software Engineering for Defence Applications: SEDA 2015 (pp. 123-131). Cham: Springer International Publishing.

[4] Sage, D., Donati, L., Soulez, F., Fortun, D., Schmit, G., Seitz, A., ... & Unser, M. (2017). DeconvolutionLab2: An open-source software for deconvolution microscopy. Methods, 115, 28-41.

[5] Titov, A., & Vukolov, A. (2019). Free and open source software for technical texts editing, its advantages and experience of usage on TMM training in Bauman University. In New Trends in Educational Activity in the Field of Mechanism and Machine Theory: 2014-2017 (pp. 208-215). Springer International Publishing.

[6] Zajdel, S., Costa, D. E., & Mili, H. (2022, September). Open source software: an approach to controlling usage and risk in application ecosystems. In Proceedings of the 26th ACM International Systems and Software Product Line Conference-Volume A (pp. 154-163).

[7] Silic, M., Back, A., & Silic, D. (2015). Taxonomy of technological risks of open source software in the enterprise adoption context. Information & computer security, 23(5), 570-583.

[8] Haider, S., Khalil, W., Al-Shamayleh, A. S., Akhunzada, A., & Gani, A. (2023). Risk Factors and Practices for the Development of Open Source Software From Developers' Perspective. IEEE Access, 11, 63333-63350.

[9] Linh, N. D., Hung, P. D., Diep, V. T., & Tung, T. D. (2019, February). Risk management in projects based on open-source software. In Proceedings of the 2019 8th International Conference on Software and Computer Applications (pp. 178-183).

[10] Schueller, W., & Wachs, J. (2024). Modeling interconnected social and technical risks in open source software ecosystems. Collective intelligence, 3(1), 26339137241231912.

[11] Harutyunyan, N. (2020). Managing your open source supply chain-why and how?. Computer, 53(6), 77-81.

[12] Nadgowda, S. (2022, November). Engram: the one security platform for modern software supply chain risks. In Proceedings of the Eighth International Workshop on Container Technologies and Container Clouds (pp. 7-12).

[13] Merigala, J., Kumar, V., Gujjarlapudi, J., Gupta, M., & Kumar, A. S. (2024, December). Analysis of Supply Chain Attacks in Open-Source Software and Mitigation Strategies. In 2024 5th International Conference on Communication, Computing & Industry 6.0 (C2I6) (pp. 1-5). IEEE.

[14] Obioha Val, O., Lawal, T., Olaniyi, O. O., Gbadebo, M. O., & Olisa, A. O. (2025). Investigating the feasibility and risks of leveraging artificial intelligence and open source intelligence to manage predictive cyber threat models. Temitope and Olaniyi, Oluwaseun Oladeji and Gbadebo, Michael Olayinka and Olisa, Anthony Obulor, Investigating the Feasibility and Risks of Leveraging Artificial Intelligence and Open Source Intelligence to Manage Predictive Cyber Threat Models (January 23, 2025).

[15] Baryannis, G., Validi, S., Dani, S., & Antoniou, G. (2019). Supply chain risk management and artificial intelligence: state of the art and future research directions. International journal of production research, 57(7), 2179-2202.

[16] Brintrup, A., Kosasih, E., Schaffer, P., Zheng, G., Demirel, G., & MacCarthy, B. L. (2024). Digital supply chain surveillance using artificial intelligence: definitions, opportunities and risks. International Journal of Production Research, 62(13), 4674-4695.

[17] Benedetti, G., Verderame, L., & Merlo, A. (2022, September). Alice in (software supply) chains: risk identification and evaluation. In International Conference on the Quality of Information and Communications Technology (pp. 281-295). Cham: Springer International Publishing.

[18] Nwamekwe, C. O., & Igbokwe, N. C. (2024). Supply Chain Risk Management: Leveraging AI for Risk Identification, Mitigation, and Resilience Planning. International Journal of Industrial Engineering, Technology & Operations Management.
[19] Röhrs, S., Rohn, S., Pfeifer, Y., & Romanova, A. (2025). Supplier Risk Assessment—A Quantitative Tool for the Identification of Reliable Suppliers to Enhance Food Safety Across the Supply Chain. Foods, 14(8), 1437.
[20] Baryannis, G., Dani, S., & Antoniou, G. (2019). Predicting supply chain risks using machine learning: The trade-off between performance and interpretability. Future Generation Computer Systems, 101, 993-1004.
[21] Shaked, A., & Margalit, O. (2022). Sustainable risk identification using formal ontologies. Algorithms, 15(9), 316.
[22] Sharma, R., Shishodia, A., Gunasekaran, A., Min, H., & Munim, Z. H. (2022). The role of artificial intelligence in supply chain management: mapping the territory. International Journal of Production Research, 60(24), 7527-7550.
[23] Li, Y., & Kong, X. (2023, August). Research on Risk Identification of New Retail Supply Chain in the Context of Internet. In 2023 2nd International Conference on Artificial Intelligence, Internet and Digital Economy (ICAID 2023) (pp. 381-389). Atlantis Press.
[24] Charles, V., Emrouznejad, A., & Gherman, T. (2023). A critical analysis of the integration of blockchain and artificial intelligence for supply chain. Annals of Operations Research, 327(1), 7-47.
[25] Mattas Konstadinos,Georgiou Pantazis,Lazaridou C. Dimitra,Mattas Christos,Nastis A. Stefanos,Seddaiu Giovanna... & Ramson Adombilla. (2025). Assessing sustainable water management in a resource-scarce environment (Ghana, West Africa) via the Analytic Hierarchy Process. Agricultural Water Management,313,109497-109497.
[26] Borko Stosic,Vladimir Djurdjević,Ivana Tošić,Antonio Samuel Alves da Silva & Tatijana Stosic. (2025). Quantifying the Rainy Season in the Brazilian Northeast Through a Modification of the Relative Entropy Method. Water,17(7),1086-1086.
[27] Jie Lu,Yuexia Zhang & Taifu Yuan. (2024). Blockchain-assisted trusted computing and communication resource allocation strategy for industrial Internet of Things. Internet of Things,27,101249-101249.