Hindawi
Scientific Programming
Volume 2021, Article ID 6691696, 11 pages
https://doi.org/10.1155/2021/6691696

Research Article

ICS Software Trust Measurement Method Based on Dynamic Length Trust Chain

Wenli Shang[1] **and Xiangyu Xing**[2,3]

[1]*School of Electronic and Communication Engineering, Guangzhou University, Guangzhou 510006, China*
[2]*Industrial Control Network and Systems Department, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China*
[3]*Information and Control Engineering Faculty, Shenyang Jianzhu University, Shenyang 110168, China*

Correspondence should be addressed to Wenli Shang; shangwl@gzhu.edu.cn

Received 15 October 2020; Revised 17 February 2021; Accepted 30 March 2021; Published 27 April 2021

Academic Editor: Ting Yang

Aiming at the real-time requirements for industrial control systems, we proposed a corresponding trust chain method for industrial control system application software and a component analysis method based on security sensitivity weights. A dynamic length trust chain structure is also proposed in this paper. Based on this, the industrial control system software integrity measurement method is constructed. Aimed at the validity of the model, a simulation attack experiment was performed, and the performance of the model was repeated from multiple perspectives to verify the performance of the method. Experiments show that this method can effectively meet the integrity measurement under the condition of high real-time performance, protect the integrity of files, and improve the software credibility of industrial control system.

1. Introduction

Industrial control system security, as an important part of the industrial control system, profoundly affects the development of industrial control network-related industries and has a strong degree of industrial relevance and industrial penetration. Information security of industrial control systems has become an important part of the integration of the two industries. The security risks it brings are no longer just "small" problems such as information leakage and unavailability of information systems. With the attack technology and means of industrial control system becoming more and more advanced, complex, and mature, the security threat of industrial control system is becoming more and more serious. Therefore, security measures are urgently needed to deal with the security threat of industrial control system. Trusted computing technology is widely concerned by industrial control industry because it can provide security immunity. As an active defense method, trusted computing can improve security of industrial control system from inside [1].

Considerable achievements have been made in the credibility of industrial control systems, but most of these achievements are based on traditional computer systems [2, 3]. Because the reliability and real-time requirements of industrial control system are not considered, these methods cannot be directly applied to industrial control system.

For the top-level design of industrial control system security, Cheng et al. [4] proposed a trusted computing-based industrial control security solution. This solution improves the security of industrial control systems through the cooperation of firewall technology, intrusion detection technology, and trusted computing technology. An et al. [5] realized the application of trusted computing technology in power system, realized higher level security protection of power system, set a precedent, and provided a case worthy of reference for the construction of security immunity engineering in other industries.

In trusted computing field, the traditional construction mode of trust chain is only applicable to a single system. For the dual redundancy system, it causes break of transitive trust. Wang et al. [6] proposed a trust chain structure based on a dual redundant system to implement credibility determination in the automatic switching process when an industrial control computer fails. Shang et al. [7–9] discussed the credibility of traditional PLC (programmable logic controller) in industrial control from different angles and achieved specific analysis of common equipment in industrial control systems.

The TCG trust chain has great potential in the construction of a trusted operating system [10], but for the measurement process from the operating system to the application, the structure of TCG trust chain is too simple to meet the diverse requirements of the application layer, and there are fewer applications in the application layer.

IMA (integrity metric architecture), as the first integrity measurement architecture based on TCG standard, is of great significance [11]. IMA determines the trustworthiness of software by performing integrity metric before the program runs, but it does not mean the trustworthiness of the program at run time. Shi et al. [12] proposed BIND (binding instructions and data), a fine-grained attestation service for securing distributed systems, which achieves small granularity dynamic verification by measuring the integrity of key code segments. However, it needs to identify key code segments and establish binding relationship between input and output data before running, which is a complex process. Shankar et al. [13] provided a largely automated system for verifying Clark-Wilson interprocess information-flow integrity. They defined a weaker version of Clark-Wilson integrity, called CW-Lite, which has the same interprocess information-flow guarantees, but which requires less filtering, only small changes to existing applications. But the formal validation capability of the model is not sufficient to formally validate the system.

Li et al. [14] proposed a dynamic trusted application model for privilege isolation, maintaining flexibility in application loading and improving the trustworthiness of the application layer. Garfinkel et al. [15] enabled the measurement of applications by using DRM (digital rights management) approach to deliver trust. Zhang et al. [16] proposed a trustworthiness analysis method for the application layer, which uses a nondisruptive behavioural approach to achieve real-time application metrics.

Industrial control system is a typical system that does not need to be restarted frequently, which makes the application of TCG trust chain in industrial control system very difficult and cannot meet the actual application requirements [17–19]. The purpose of building a trusted operating system is to ensure that the application has a trusted execution environment and that the trust relationship can continue. For industrial control systems, based on meeting the reliability and real-time requirements of industrial control systems, improving the credibility of application programs is a very critical issue [20, 21]. In this paper, we start with the static measurement method, focus on real-time, and build a trusted approach at the application layer. Compared with the existing literature, this paper has the following major contributions:

(1) A software component analysis method is proposed that takes security sensitivities into account. This approach enables a hierarchical protection scheme for different files by dividing the software components by considering the security sensitivity of the files. The solution can effectively improve the real-time performance of applications when performing integrity checks.

(2) A new dynamic length chain of trust transfer model is proposed. The model can be adapted to different real-time requirements and complete integrity verification on the basis of meeting the real-time requirements of industrial control systems. This new dynamic length chain of trust structure has the features of easy updating, dynamic structure, and adjustable real-time requirements.

(3) A software trustworthiness verification model based on dynamic length chains of trust is proposed. Adopt the concept of differentiated design of the application software's own components and the system's common components to achieve classification and management of different components and to optimise the efficiency of integrity verification. Effectively improve the efficiency of system integrity verification, reduce the burden of trusted computing on the system so that trusted computing technology means can continue to be used in the absence of system resources and can be effectively compatible with the old industrial control system, and enhance the system's trustworthiness as much as possible at a lower upgrade cost. The update process of the model is also discussed, which is characterised by easy software updates.

The rest of this paper is organized as follows. ICS software analysis is discussed in Section 2. A software trust measurement model based on dynamic length trust chain (DLTC) is proposed in Sections 3. The effectiveness of the algorithm is verified by a simulation experiment in Section 4. Finally, the conclusions are given in Section 5.

2. ICS Software Analysis

2.1. Real-Time. Industrial control system is facing a large number of new security challenges, and industrial control computer is undertaking more and more information security responsibilities. Information security protection methods generally adopt the combination of active defense and passive defense. As an important technology of active defense, trusted computing is becoming more and more important. Generally, the combination of dynamic and static methods is used to ensure the credibility of the system. Integrity measurement is a common static measurement method. However, it also has an inherent disadvantage: integrity measurement of trusted computing is usually performed before the program runs. For services with real-

time requirements, careful consideration must be given to whether to use static measurement methods for integrity measurement [22].

Real-time means that the input, calculation, and output of the signal are completed in a very short time and processed in time according to the changes in the generation process. Real-time is the ability to perform specified functions within a limited time and respond to external asynchronous events. Emphasis is on the specified time, as long as the completion within the specified time is real-time [23].

Specifically, for any stimulus-response system, there is a time from the stimulus input to the response output, that is, the stimulus-response period T, which represents the time response capability of the system.

If the response time T of the system can meet the requirement of the response time t specified by the system, that is, $t \leq T$, the system is a real-time system.

In industrial control systems, two main factors are affecting the real-time performance of the system: on the one hand, the real-time nature of the unit components in the system. That is, controllers, sensors, and actuators must meet real-time requirements. On the other hand, it refers to the real-time nature of the industrial communication network, and the information interaction between field devices must be completed within a certain time.

According to the requirements of different systems for real-time requirements, field information can be divided into real-time information and nonreal-time information. Real-time information must be processed on time, with high requirements for real-time performance. Priority must be given to prevent system failure. At the same time, the credibility of this operation is higher. The execution of this part of the program is related to the credibility of the system's fault handling behavior.

The integrity measurement method usually increases the delay and affects the real-time performance of the system. Therefore, for systems with high real-time requirements, integrity is to measure integrity on the premise of meeting real-time requirements.

Integrity measurement and real-time are usually mutually limited. For hard real-time systems, the delay fluctuation in integrity measurement may have a significant impact, resulting in that system cannot operate normally. For soft real-time system, the impact of integrity measurement is less than that of hard real-time system. At the same time, the availability of industrial control system is very important. There is a strong positive correlation between real-time and availability. Generally speaking, availability and real-time should be considered first when considering the security of industrial control system.

2.2. File Type. Documents have different security sensitivities and different levels of security protection. For security-sensitive files, more security measures need to be added to ensure that they are secure. For security-sensitive files, such as dynamic link library, static link library, and Perl script, this paper analyzes the file characteristics of Linux platform and the emphasis of integrity measurement.

This paper mainly analyzes and studies the common file types in Linux system. For example, the binary file is the traditional executable file, while the script language program is generally described as a text file.

This paper considers that security-sensitive files under the Linux platform which can be divided into the following three categories: the first category is executable files, which includes the files that can be executed with executable permission, and the files that can be executed after getting executable permission but have no executable permission at present. These files include the main files that can be directly run, including dynamic link libraries, static link libraries, binary files, and scripts. The second category is the files containing sensitive data. These files store information that may be used during program operation, such as files that record the hash value of files. They need to be verified for integrity to prevent hackers from modifying it after obtaining administrator rights of these files. The third category is the user-defined security-sensitive files. Users can define files as security-sensitive files according to actual needs.

2.3. Component Analysis Method Based on Security Sensitivity. The previous paper analyses the software requirements and characteristics of industrial control systems from three perspectives. To better describe the internal connections and characteristics of these files, a two-dimensional attribute is constructed to describe the characteristics of the files. Through cluster analysis of the file characteristics, the internal connections of the files are obtained, and the files are decomposed to meet the needs of establishing a real-time trust chain structure.

Software attributes are constructed from two dimensions: the first is the weight of security sensitivity. Different levels of security sensitive files need different weights, which need to be determined artificially and protected by different protection levels. This is a "hierarchical protection" method [24], which can effectively deal with complex practical situations.

The second is the relationship and function between calling and called. The calling relationship in a program is usually organized by function. The granularity of analyzing software behavior based on function is too small, which leads to too many details involved in measurement. Usually, the number of files involved in a program that performs a specific function is fixed, and not all files are involved. By analyzing the calling relationship of a specific function, the number of files that need to be verified when using the function can be effectively reduced, so as to improve the real-time performance of integrity measurement.

Due to the difference of software structure, it is difficult to analyze the calling and called functions with automatic method, and it depends more on experience. Taking OpenPLC as an example, it is mainly based on the inclusion of header files, and the calling relationship of program files is determined by the operation of header files. OpenPLC file structure is clear and easy to identify. Usually, each folder is a file that performs independent functions.

Through the above two attributes, different files can be classified and processed. The software is written with different software structure, and the analysis is slightly different. A well-designed software should meet the requirements of "high cohesion and low coupling." For such software, it is easy to handle at the file and folder level.

According to these two principles, files with similar functions are put in the same "package." The package consists of several files with similar functions and security sensitivity [25]. These files should show the relationship between the calling and the called. The concept of package essentially defines the delay of component, but the granularity of the component is different from the traditional definition [26, 27]. This method puts more emphasis on the security sensitivity of files, which is an improvement of component definition.

2.4. Dynamic Length Trust Chain. In the industrial control system, the software update is slow, and it takes a lot of time to realize the dynamic measurement method in the industrial control system [28]. At present, static method is widely used because of its simple and easy deployment [29].

Generally speaking, these methods are easy to implement and highly customized, and traditional models use TPM as the source of trust. As a trusted root, TPM has been proved to be a feasible solution and has been widely used. Many scholars put forward the construction technology of virtual trusted root in cloud environment [30–32]. For some industrial control systems without TPM, the trusted root based on USB can be used, which can also bring the expected effect [33, 34].

The trust chain length of TCG organization is fixed, and the measurement time is relatively fixed, which cannot meet the real-time requirements of industrial control system. If we give up the measurement of some documents directly, it is difficult to guarantee their credibility. Therefore, we need a dynamic length of trust chain, and cut the length of the trust chain to meet the requirements of trust degree. At the same time, the chain trust transfer model cannot effectively describe the call and transfer of control between applications. Even for entities with measurement capability, with the increase of the number of entities in the computing platform, the chain trust transfer model will become difficult to manage.

In view of these shortcomings, this paper proposes a Dynamic length trust chain (DLTC) structure for real-time industrial control systems. This structure effectively solves the problems of fixed length of trust chain, fixed measurement time, and poor real-time performance. At the same time, this structure can also effectively describe the calling relationship between software. The description of software dependency can adapt to the current software environment.

The file package sequence obtained by using the analysis method based on security sensitivity and components is $F_0, F_1, F_2, \ldots, F_n$, recorded as Ω, as shown in (1). F_0 is the main function; n is the number of file package; there are n+1 file packages in Ω:

$$\Omega = \{F_0, F_1, F_2, \ldots, F_n\}. \tag{1}$$

For a program, the description of its package is shown in (1), and then a corresponding chain of trust can be described as an ordered sequence as described in (2); $\varphi i = 1$ means the file package is in TL; $\varphi i = 0$ means the file package is not in TL; in particular, $\varphi 0 = 1$ means the trust chain always contains the main function.

$$\mathrm{TL} = \{\varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n\}. \tag{2}$$

The calculation method of the length definition of the trust chain is shown in

$$\mathrm{len} = \sum_{i=0}^{n} \varphi_i. \tag{3}$$

Equations (1) and (2) describe the elements contained in the trust chain. In essence, the trust chain is still a chain structure trust chain, but because the successor nodes of each node in the trust chain are dynamically determined, it forms a tree structure trust chain.

3. Software Trust Measurement Model Based on DLTC

3.1. Real-Time Mathematical Description. For an entity, the system response time requirement is T, which means that the response time [35] meets the following equation:

$$t \le T. \tag{4}$$

For an entity, the response time without adding the integrity measurement method is t_1, and the time required for its integrity measurement is t_2, and then the total response time satisfies the following equation:

$$t = t_1 + t_2. \tag{5}$$

Then, the expected maximum response time of the chain of trust is defined, which is as shown in equation (6). In equation (6), α is the dynamic coefficient, and $0 \le \alpha \le 1$. Its existence is to leave a margin for the estimation error of the system in the required measurement time, so as to ensure that the system will not fail due to exceeding the response time requirement under hard real-time conditions.

$$T_{h\max} = \alpha \times (T - t_1). \tag{6}$$

For any package F_i $(1 \le i \le n)$, the files in it are marked as f_{ij}. There are two attributes for anyone, one is the file size, marked as s_{ij}; the other is the expected measurement time of the file, marked as τ_{ij}, which describes the end of the comparison of the completeness of the hash calculation value integrity check result of the file.

The expected measurement time τ_{ij} for any file satisfies the following equation:

$$\tau_{ij} = a + b \times s_{ij}. \tag{7}$$

The calculation of equation (7) is obtained by least square fitting, which describes a relationship. The time

required for a file to perform integrity measurement consists of two parts. One is the file I/O time, and the other is the time required for hash calculation. In equation (7), a describes the fixed overhead time of file I/O, and b describes the measurement time that increases as the file size increases. This part of the time is mainly generated by the hash operation.

The actual values of the parameters a and b need to be calculated according to the actual configuration of the system. The calculation process of this parameter in the experimental environment of this paper is detailed in Section 4.2.

Then, the integrity measurement time of each file package is $\sum_j \tau_{ij}$, which is defined as the measurement time sum of all files contained in the file package.

The most time-consuming step in the dynamic generation of a complete trust chain is the hash operation, but other smooth processing also consumes a lot of time. The main steps of trusted chain generation include the following parts: (1) Due to the serial original in the TPM, the backlog of unfinished operations will affect the subsequent operations, which will cause delays, which is recorded as t_{TPM}, as shown in Figure 1. (2) Processing hash metric list (HML) operations requires time. HML is defined in Section 3.2. This part of the time can be divided into HML reading time, recorded as $t_{\mathrm{HML}r}$, and time to write HML, recorded as $t_{\mathrm{HML}w}$. (3) The time consumed to generate the random number, process the random number, and generate the chain of trust is t_{randm}. (4) The time to calculate the integrity of all the files in the chain of trust.

$\mathrm{T}_{h\ \max}$ can also be described in the following equation:

$$T_{h\max} = t_{\mathrm{TPM}} + t_{\mathrm{HML}r} + t_{\mathrm{randm}} + \sum_{i=0}^{n} \mathrm{TL}(i) \times t_{Fi}. \quad (8)$$

3.2. Hash Metric List. For a program, its related information is recorded in a text file, called a hash metric record table, abbreviated as Hash Metric List (HML). It is stored in a trusted storage space controlled by the TPM, it is encrypted by the encryption algorithm contained in the TPM, and the key is stored in the TPM to ensure the security of the HML. The file structure of the HML is shown in Figure 2.

The main information stored in HML includes the following contents: the file structure of the software and all files, the number of measurements transferred by each file, total number of measurements, the file size, and the integrity measurement results of each file.

In order to prevent the HML file from being tampered, integrity measurement and report should be carried out before using the HML file every time to determine the credibility of the HML file.

To maintain HML more conveniently and efficiently, we divide HML file into two types, the first one is a system hash metric record table, denoted as SHML (system hash metric list), and the second one is an application hash metric record table, denoted as AHML (application hash metric list). SHML is used to store the hash value measurement results of public resources in the system, and AHML is used to record the hash value measurement results of the application's files.

The main reasons for adopting this design method are as follows:

(1) Storing all records in a unified HML will lead to too large HML file and low search efficiency, which cannot meet the design goal. The use of large HML files will have a great impact on the real-time performance of the system.

(2) The component-based design method produces a large number of public resources. Many of them are public resources in the system as dynamic link library (DLL). A large number of services call DLL. If each HML stores relevant information, it will cause a great waste of resources.

(3) Credibility of integrity measurement results can be considered as short-term rather than long-term, or invalid after measurement. Whether two integrity measures are needed in two calls to common resources in a short time is worth discussing. By considering the validity period of integrity measurement, we can effectively reduce the time of integrity measurement and improve the real-time performance of the system.

(4) Consider the security principles of "least privilege" and "as needed." These DLL modules only serve specific programs. If they are all stored in a unified HML file, it may bring information security risks.

In summary, the HML is divided into a system-maintained SHML for controlling common components and an application-specific AHML.

As described in Section 2.3, through the analysis of software components, the files in a complete software are classified, and the loose file structure is changed into a compact file package structure. The granularity of the package can be adjusted according to the measurement requirements. AHML is dynamically formed in the trust chain constructed in this paper. To ensure information security, two files are used to manage AHML, AHML-H, and AHML-F.

AHML-H is used to store software packages, absolute path names and integrity metrics. It stores sensitive information. In order to prevent tampering, it separates the information that needs to be modified from the information that does not need to be modified during the operation. AHML-H stores the integrity measurement results and encrypts them with the encryption function provided by TPM.

AHML-F is used to store packages. The main information it stores is the file package, package size, measurement times, measurement interval, security sensitivity weight, and total measurement times. The file is dynamic when the trust chain is generated. In order not to store sensitive check value information and ensure security, the random number selection algorithm is used to achieve the measurement times, so as to achieve better robustness and ensure the correct operation of the program when the measurement times have problems.

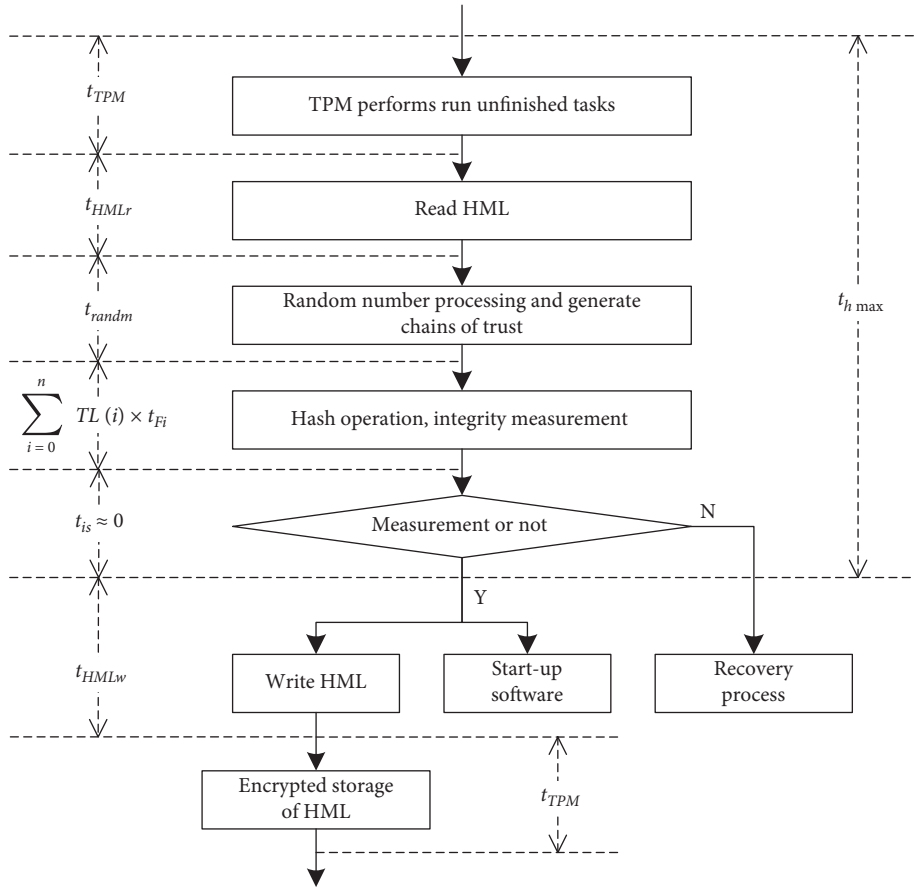

Figure 1: Trust chain generation process and time consumption.

HML
Strategy developers: ·····
Digital certificates: ·····
Digital signatures: ·····
Total number of measurements: ·····
Id; file-package; absolute path filenames; size; integrity metrics; number of measurements;

Figure 2: File structure of the HML.

Implementing AHML as two files has the following advantages: firstly, the separation of sensitive information and nonsensitive information is realized. AHML-H and AHML-F are stored in trusted storage areas. There is no sensitive information in AHML-F, and it will be modified dynamically during the running of the program to ensure its security. Once the memory leak occurs, the impact will be very small. AHML-H file stores sensitive integrity measurement results, which can ensure its security by only reading and not modifying.

Secondly, unnecessary information can be hided. AHML-H file itself does not reflect those files that need to be measured, which can reduce the time of file path and integrity measurement results appearing in memory and prevent the information security problem that the software cannot be loaded into memory when using static measurement method.

Finally, the size of related files can be reduced, and the complexity of AHML-H file retrieval can be reduced. It can save time to measure while retrieving. By separating reading AHML-H file from writing AHML-H file, the writing operation of AHML file is independent of measurement, which can save measurement time.

3.3. Trust Chain Generation Method Based on Roulette Rules. For a software, a series of packages are obtained by using the "analysis method based on security sensitivity and component," which maintain an AHML-F file and an AHML-H file. The operating system maintains a SHML file. For AHML-F, the data item "measures" constitutes an array M of lengths 1×n, where M_i is number of times the i-th file package was measured. The combination of arrays M and

random numbers determines the generation of trust chains. The random number is generated by the TPM. The random number generator is one of the several functions provided by the TPM. Its data item "security sensitivity weight" constitutes an array W of lengths 1×n, where W_i is the weight of the i-th file package.

Each time TPM generates a random number R, the mapping between the random number and the file package is generated by the roulette selection method. The steps of generating trust chain file by roulette selection method are as follows.

3.3.1. Handling Arrays M and Arrays W. The array W exists in the form of characters and needs to be converted into numeric values. The array W reflects the adjustment of the trust chain structure according to the security sensitivity of the package, which affects the structure of the trust chain in a proportional way. Depending on the sensitivity, the values are 4, 3, 1, and 0.5.

To improve the robustness of the algorithm, restrictions are added. For packages that meet the conditions shown in equation (9), in the selection of trust chain, the probability of being selected is 1, where max-() means calculating the maximum value of the array. k is the limited number of times, which can be adjusted according to the actual situation. In this paper, k = 5.

$$\max(M) - M_i > k. \tag{9}$$

Equation (9) is to ensure that the difference between the number of times any two packages are measured is not greater than 5, so as to ensure that in extreme cases, non-security sensitive data can also get a limited number of integrity measures. The robustness of the algorithm is improved.

3.3.2. Selection Probability. Roulette selection method, also known as proportional selection method, is based on the proportion of individuals in the whole. First, the array M is updated by the following equation:

$$M_i = \max(M) - M_i + 1. \tag{10}$$

Then, the probability of selection is calculated by the following equation:

$$P(F_i) = \frac{M_i \times W_i}{\sum_{i=1}^{n} M_i \times W_i}. \tag{11}$$

3.3.3. Cumulative Probability. Selection probability of each package is calculated by the following equation:

$$q_i = \sum_{j=1}^{i} P(F_i). \tag{12}$$

3.3.4. Random Number Selection. The random number R generated by the TPM is used to determine the file package to be selected. If $R <$ q [1], select package 1; if q[h-1]$< R <$ q [h], select the h-th individual.

3.3.5. Repeating the Above Process until Sufficient Data Are Generated. In this way, the trust chain structure can be generated dynamically. Due to the robustness of the algorithm, the measurement times of the algorithm will not be greatly different.

3.4. Trust Chain Update Mechanism. The trust chain of TCG chain structure adopts the extension operation of hash, which makes it difficult to update. Once the hash value of a node is updated, the trust chain needs to recalculate the hash value and carry out the extension operation, which leads to a lot of calculation work. To solve this problem, the star trust chain stores hash values for different nodes, which can effectively solve the problem of updating. The trust chain structure proposed in this paper has the characteristics of star structure trust chain and is easy to update.

The update process of file modification is as follows:

(1) Update AMHL-H file. Update the files that need to be updated and modify the corresponding records in the AMHL-H file.

(2) Update the AHML-F file. Update the file size of the corresponding package.

The update process of file addition is as follows:

(1) Calculate the distance between the feature description of the new files and the cluster center; classify the new files to the closest file package.

(2) Update AMHL-H file. Add new records to the AMHL-H file.

(3) Update the AHML-F file. Update the file size of the corresponding package.

The update process of file deletion is as follows:

(1) Update AMHL-H file. Delete related records located in AMHL-H files.

(2) Update the AHML-F file. Update the file size of the corresponding package.

4. Experiment

4.1. Simulation Platform Building. The configuration of the experimental platform is shown in Table 1. Due to the limitation of experimental conditions, TPM v1.2 is adopted. Theoretically, all functions implemented on TPM v1.2 can be implemented on TPM 2.x [36].

This experiment only verifies the effectiveness, and the performance experiment only represents the running effect in the current experimental environment. The benchmark data measured in Section 4.2 only represent the current experimental platform.

4.2. Benchmark Data Measurement. In order to objectively measure the impact of device configuration on the time

Table 1: Configuration information for the prototype system.

Name	Version
System	Ubuntu 18.0
CPU	Core i5-4200 M 2.50 GHz, 4 core
Memory	4 GB
Secondary cache	4 MB
TPM version	1.2
Gun multiple precision arithmetic	6.1.2
Trousers	0.3.14
TPM-tools	1.3.9.1
TPM chip version	1.2.0.7
TPM spec level	2
TPM errata revision	1
TPM vendor ID	ETHZ
TPM version	01010000
TPM manufacturer info	4554485a

consumption of TPM hash operation, the following experimental scheme is used to test:

(1) Randomly generate 1024 files, the file size is from 1 KB to 1024 KB, and the interval is 1 KB.

(2) Hash these files and extend them to PCR through extend operation, and record the time required for each operation, the unit of measurement is *ms*, and measure 100 times repeatedly.

(3) Processing data: remove the data less than the smaller quartile, remove the data greater than the larger quartile, and calculate the arithmetic mean of the remaining data to replace the whole data.

(4) The data calculated in step 3 are fitted by least square method, and the fitting calculation is carried out by

$$T_{\text{hash}} = 0.0022 \times F_s + 0.0621, \tag{13}$$

where T_{hash} is the time required to hash the file, and the unit of measurement is ms; F_s is the size of the file to be measured; the unit of measurement is KB.

The determination coefficient of the fitting equation is $r^2 = 0.9989$, which indicates that the fitting result is good.

4.3. Case Analysis Based on OpenPLC. OpenPLC is an easy-to-use programmable logic controller based on open-source protocols [37, 38]. For OpenPLC, the first dimension of its file attributes is analyzed in two steps:

Step 1: filter the installation file name and authority.

Through the LS command in Linux, output all the file names and related authorities, and filter out the files that can be considered as the security-sensitivity weight of D through reverse filtering. Files with such characteristics are usually as follows:

(1) C/C++ language source code, Java source code, Makefile and other files with source code: script files are not included. For OpenPLC, these files are only used in the installation, and do not work in the subsequent software operation.

(2) Config files used during software installation: these files serve Linux software installation. Config file used in the software running process is not included.

(3) Intermediate files in the compilation process: there are a lot of C/C++ programs in OpenPLC, which will produce some intermediate files in the process of compiling.

(4) Auxiliary function files: these files are used to assist users and have nothing to do with the trusted operation of the software. These files mainly include help documents, readme, installation logs, and copyright license files.

(5) Files unrelated to the installation platform: in OpenPLC, there are related files for Windows platform and docker container. Whether these files have security sensitivity depends on different experimental platforms. For the platform used in this experiment, these files are useless.

(6) When the software has strong robustness, some image files can be considered as not security-sensitive. The important premise is that the software has enough robustness and will not fail because of the lack of these image files.

Step 2: through manual measurement, security sensitivity of files can be determined more accurately.

For OpenPLC, some folders are used to store software output files. These are software outputs and have nothing to do with the operation of the software. The security of these outputs can be guaranteed by data encryption, which is beyond the scope of this article.

According to the above method, the OpenPLC package number and file size are shown in Table 2. As shown in Table 2, the total size of the file package is 1092.5 KB. According to (13), the expected measurement time is 24.0961 ms, the maximum file package size is 861.4 KB, and the expected measurement time is 18.5721 ms, accounting for 77% of the total expected measurement time. The main reason for this problem is that the file package contains files with large memory consumption.

4.4. Effectiveness Analysis. After successful installation of the experimental environment, comparative experiments are designed to verify the effectiveness of the algorithm. The validation of the algorithm mainly considers whether the algorithm can effectively prevent illegal start-up behavior.

Table 3 shows the validity of the algorithm. The above analysis shows that the algorithm is effective.

4.5. Performance Analysis. Dynamic length trust chain mainly solves the strict real-time requirements of industrial control system, so the fluctuation of its performance is very important for the credibility of industrial control system software and the performance analysis of model. Real-time

Table 2: Package number and size.

No.	Directory	Size	Security sensitivity
0	./start_openplc.sh	46	4
1	./utils/dnp3_src/	182300	4
2	./utils/libmodbus_src	40139	4
3	./utils/matiec_src	1161242	4
4	./utils/st_optimizer_src	32136	4
5	./webserver/otherfile	8615563	4
6	./webserver/static	538565	1
7	./webserver/core	396202	3
8	./webserver/lib	213255	3
9	./webserver/scripts	7303	3

Table 3: Algorithm validation results.

Test case	Use integrity measurement	Do not use integrity measurement
Use correct start_openplc.sh file	Successful	Successful
Modify one of the security-sensitive files	Unsuccessful	Successful
Modify the file bool_true.png that does not affect the application	Successful	Successful
Update libmodbus.o with security measures	Successful	Successful
Infect files with a virus bootkit [39–41]	Unsuccessful	Successful

analysis can be done from two aspects: low delay requirement and high delay requirement.

First, the test is executed when the real-time requirement is low, all metrics can be executed, and the expected maximum response time is limited. This time limit can fully meet the package distribution shown in Table 2. Under this real-time condition, the experimental data of 100 repeated tests are shown in Figure 3.

In the data shown in Figure 3, the maximum consumption time is 25.6791 ms, the minimum consumption time is 23.0454 ms, the average consumption time is 24.2340 ms, and the variance is 0.3627. From the data fluctuation and variance data shown in Figure 3, we can see that the stability of the algorithm is very good, and the data fluctuation is small. In the process of integrity measurement, the expected maximum response time $T_{h\ \max}$ will not be exceeded, which can meet the requirements of low real-time.

Secondly, test is performed for a situation that the real-time requirement is high and a large number of measurements cannot be carried out. In this case, the scheduling ability of the algorithm determines the number of times the file is measured. The limit dynamic coefficient is $\alpha = 0.9$, the maximum expected response time is $T_{h\ \max} = 24$ ms, and the time required for OpenPLC to perform a hash operation is $T0 = 24.0961$ ms, here $T0 > T_{h\ \max}$. At the same time, considering the fluctuation of calculation time in the measurement process, the integrity of all files cannot be calculated in the measurement process. In order to meet the real-time and robust requirements of industrial control system, the performance analysis of the algorithm should be carried out from two aspects: (1) considering the fluctuation of the algorithm. The fluctuation of algorithm will lead to timeout and destroy the availability of industrial control system; (2) considering the fluctuation of packet measurement process. The number of times each software package is measured should be as stable as possible to ensure that each file has the opportunity to be measured, so as to improve the credibility of the whole software system.

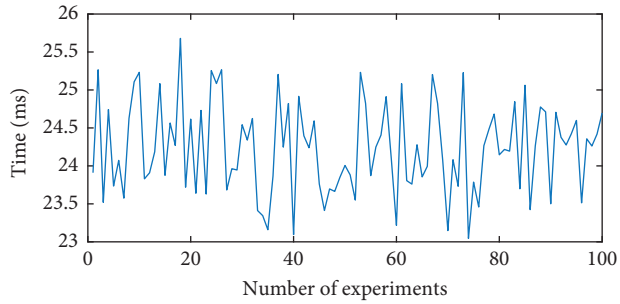

Figure 3: Running time during low real-time request.

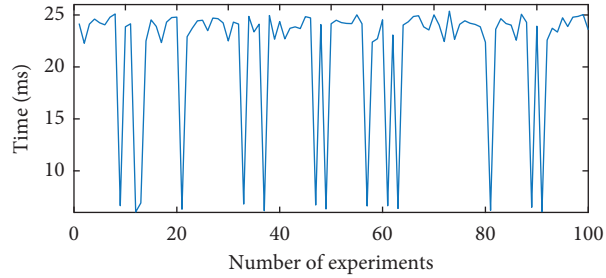

Figure 4: Running time ($T_{h\ \max} = 24$ ms).

In Figure 4, the maximum value is 25.3655, and 54% of them is larger than $T_{h\ \max}$. All samples are less than $T_{h\ \max\alpha}$. $\alpha = 0.9$ is used to describe the margin left by the algorithm. It can be seen that the algorithm does not exceed the limit. The results show that the fluctuation of the algorithm is reasonable and will not cause the program execution to fail beyond the time limit.

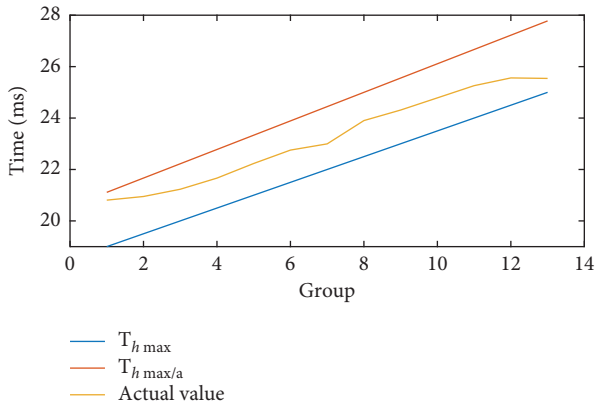

Figure 5: Maximum running time.

When dynamic coefficient $\alpha = 0.9$ and the maximum distribution of the expected maximum response time $25\,\text{ms} \geq T_{h\ max} \geq 19\,\text{ms}$, distribution of the maximum running time is shown in Figure 5.

5. Conclusions

According to the real-time requirements of industrial control system, as well as the operating system and application software rarely involved in TCG related research, the corresponding trust chain construction method is proposed in this paper. According to the characteristics of industrial control system software, a component analysis method based on security sensitivity weight is proposed from three aspects of real-time requirements, component characteristics, and file characteristics. Based on the analysis of traditional trust chain structure, a dynamic length trust chain structure is proposed. The generation process of this trust chain structure is described in detail, and an example is analyzed.

The effectiveness of the model is evaluated from two aspects. The effectiveness of the model is verified by simulation attack experiments, and the effectiveness of dynamic length trust chain is analyzed by simulating the impact of different attack behaviors on file changes. Experiments show that this method can effectively deal with various attacks, protect the integrity of the file, and improve the credibility of the program. The performance of the model is analyzed by repeated experiments. Performance analysis experiments show that the method can meet different real-time requirements.

This paper studies the measurement method of operating system and application in trust chain and proposes a new trust chain structure. However, this method still needs to be further improved. At present, the degree of automation of the analysis process is low, which requires the intervention of human experience, and the process is more complex. The real-time condition is limited, because information security technology will inevitably lead to delay, and the harsh real-time requirements cannot be met. These are our further research directions.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the "National Key R&D Program of China" (2018YFB2004200), the Open Project of Zhejiang Lab "Construction Technology of Local High Security Trusted Execution Environment for Edge Intelligent Controller" (2021KF0AB06), and the National Natural Science Foundation of China "Research on anomaly detection and security awareness method for industrial communication behaviours" (61773368). The authors would also like to acknowledge the helpful comments and suggestions of the Industry Control System Security Software Group. Their efforts are greatly appreciated.

References

[1] R. R. Schell and M. F. Thompson, "Platform security: what is lacking?" *Information Security Technical Report*, vol. 5, no. 1, pp. 26–41, 2000.

[2] T. Morris, R. Vaughn, and Y. Dandass, "A retrofit network intrusion detection system for MODBUS RTU and ASCII industrial control systems," in *Proceedings of the 2012 45th Hawaii International Conference On System Sciences*, pp. 2338–2345, IEEE, Maui, HI, USA, January 2012.

[3] S. L. P. Yasakethu and J. Jiang, "Intrusion Detection via Machine Learning for SCADA System Protection," in *Proceedings of the 1st International Symposium For ICS & SCADA Cyber Security Research*, pp. 101–105, Guildford, Leicester, UK, September 2013.

[4] S. Cheng and Z. Lianggao, "An information security solution scheme of industrial control system based on trusted computing," *Information And Control, China*, vol. 44, no. 5, pp. 628–640, 2015, in Chinese.

[5] N. Y. An, Z. H. Wang, and B. H. Zhao, "Research and application of trusted computing in electric power system," *Journal of Information Securyity Research*, vol. 3, no. 4, pp. 353–358, 2017, in Chinese.

[6] L. Wang, M. H. Yang, Z. L. Liu, and J. Q. Zheng, "Trust chain generating and updating algorithm for dual redundancy system," *Journal on Communications*, vol. 38, no. 1, pp. 1–8, 2017, in Chinese.

[7] W. L. Shang, X. Y. Xing, X. D. Liu, L. Yin, and E. Y. Gao, "Research on security enhancement of total shipcomputing environment based on trusted computing," *Ship Science and Technology*, vol. 42, no. 13, pp. 125–129, 2020, in Chinese.

[8] W. L. Shang, L. Yin, X. D. Liu, and J. M. Zhao, "Construction technology and application of industrial control system security and trusted environment," *Netinfo Security*, vol. 6, pp. 1–10, 2019, in Chinese.

[9] M. J. Li, L. D. Wang, W. Xiong, and H. Ding, "Research on trusted computing constructing technology for PLC system," *Software Guide*, vol. 16, no. 11, pp. 168–175, 2017, in Chinese.

[10] C. Li, R. Li, and L. Zhuang, "Formal analysis of trust chain," in *Proceedings of the Second International Conference on Networks Security*, pp. 111–116, Wireless Communications and Trusted Computing, Madurai, India, July 2010.

[11] R. Sailer, X. Zhang, T. Jaeger, and L. Van Doorn, "Design and implementation of a TCG-based integrity measurement

architecture," in *Proceedings of the USENIX Security Symposium*, pp. 223–238, San Diego, CA, August 2004.

[12] E. Shi, A. Perrig, and L. Van Doorn, "Bind: a fine-grained attestation service for secure distributed systems,," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P'05)*, pp. 154–168, IEEE, Oakland, CA, USA, July 2005.

[13] U. Shankar, T. Jaeger, and R. Sailer, "Toward automated information-flow integrity verification for security-critical applications," in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2006*, San Diego, California, USA, October 2006.

[14] X. Y. Li and C. X. Shen, "Research to a dynamic application transitive trust model," *Journal of Huazhong University of Science and Technology (nature Science)*, vol. 33pp. 310–312, z1, 2005, in Chinese.

[15] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh, "Terra," *ACM SIGOPS Operating Systems Review* in *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, vol. 37, no. 5, pp. 193–206, Bolton Landing, NY USA, Octobor 2003.

[16] F. Zhang, M. D. Xu, H. C. Chao et al., "Real-time trust measurement of software: behavior trust analysis approach based on noninterference," *Journal of Software*, vol. 30, no. 8, pp. 2268–2286, 2019, in Chinese.

[17] X. Y. Li, Z. Han, and C. X. Shen, "Transitive trust and performance analysis in Windows environment," *Journal of Computer Research and Development*, vol. 44, no. 11, pp. 1889–1895, 2008, in Chinese.

[18] T. Huang and C. X. Shen, "A trusted bootstrap solution based on a trusted server," vol. A01, pp. 12–14, Journal of Wuhan University, 2004, in Chinese.

[19] L. Yan, J. Zhang, and A. Zhang, "Scheme of trusted bootstrap based on general smart card," *Journal of Beijing University of Technology*, vol. 43, no. 1, pp. 100–107, 2017, in Chinese.

[20] L. H. Fu and D. Wang, "Research on Trust Evaluation of Secure Bootstrap in Trusted Computing Based on Fuzzy Set Theory," in *Proceedings of the 2010 International Conference On Machine Learning And Cybernetics*, pp. 592–595, IEEE, Qingdao, China, July 2010.

[21] K. Balasubramanian and A. M. Abba, *Secure Bootstrapping Using the Trusted Platform Module*, IGI Global, Pennsylvania, USA, 2018.

[22] L. Davi, A. R. Sadeghi, and M. Winandy, "Dynamic integrity measurement and attestation: towards defense against return-oriented programming attacks," in *Proceedings of the 2009 ACM Workshop on Scalable Trusted Computing*, pp. 49–54, ACM, Chicago Illinois, USA, November 2009.

[23] Z. Quan, X. Yuan, and Y. Zhu, "Real-time flow control system based on siemens PLC," in *Proceedings of the 2019 IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 1703–1708, IEEE, Tianjin, China, August 2019.

[24] C. Weiping, "The application of trusted computing 3.0 in classified protection standard system 2.0," *Journal of Information Security Research*, vol. 4, no. 7, pp. 633–638, 2018, in Chinese.

[25] Y. Yu, Y. G. Liu, and J. Gu, "Analysis and measurement of components trust relationship in internetware system," *Netinfo Security*, vol. 18, no. 1, pp. 31–37, 2018, in Chinese.

[26] G. J. Holzmann, "Software components," *IEEE Software*, vol. 35, no. 3, pp. 80–82, 2018.

[27] B. Wang, Y. Chen, S. Zhang, and H. Wu, "Updating model of software component trustworthiness based on users feedback," *IEEE Access*, vol. 7, pp. 60199–60205, 2019.

[28] T. Suzuki, *TPM in Process Industries*, Routledge, UK, London, 2017.

[29] C. Shen, H. Zhang, H. Wang et al., "Research on trusted computing and its development," *Science China Information Sciences*, vol. 53, no. 3, pp. 405–433, 2010, in Chinese.

[30] M. Chiregi and N. J. Navimipour, "Trusted services identification in the cloud environment using the topological metrics," *Karbala International Journal of Modern Science*, vol. 2, no. 3, pp. 203–210, 2016.

[31] R. Shaikh and M. Sasikumar, "Trust model for measuring security strength of cloud computing service," *Procedia Computer Science*, vol. 45, pp. 380–389, 2015.

[32] M. Chiregi and N. Jafari Navimipour, "Cloud computing and trust evaluation: a systematic literature review of the state-of-the-art mechanisms," *Journal of Electrical Systems and Information Technology*, vol. 5, no. 3, pp. 608–622, 2018.

[33] Y. Hu and H. Lv, "Design of Trusted BIOS in UEFI Base on USBKEY," in *Proceedings of the 2011 International Conference On Intelligence Science And Information Engineering*, pp. 164–166, IEEE, Wuhan, China, August 2011.

[34] D. Zhang, Z. Han, and G. Yan, "A portable TPM based on USB key," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, pp. 750–752, ACM, Chicago, Illinois, USA, October 2010.

[35] B. M. Wilamowski and J. D. Irwin, *Industrial Communication Systems*, "CRC Press, Boca Raton, Florida, USA, 2018.

[36] Trusted Computing Group: TCG Specification Architecture Overview, 2007, https://trustedcomputinggroup.org/wp-content/uploads/TCG_1_4_Architecture_Overview.pdf.

[37] T. Alves and T. Morris, "OpenPLC: an IEC 61,131-3 compliant open source industrial controller for cyber security research," *Computers & Security*, vol. 78, pp. 364–379, 2018.

[38] S. Fujita, K. Hata, and A. Mochizuki, "Open-PLC based control system testbed for PLC whitelisting system," *Artificial Life and Robotics*, vol. 26, no. 4, pp. 1–6, 2020.

[39] W. Showalter, "A universal Windows bootkit: an analysis of the MBR bootkit HDRoot," in *Proceedings Of the 50th Hawaii International Conference On System Sciences*, pp. 6060–6068, Hawaii, USA, January 2017.

[40] H. Gao, Q. Li, Z. Yu et al., "Research on the working mechanism of Bootkit," in *Proceedings Of the 2012 8th International Conference on Information Science and Digital Content Technology (ICIDT2012)*, pp. 476–479, IEEE, Jeju, South Korea, June 2012.

[41] H. J. Hu, M. Y. Fan, and G. W. Wang, "Concealment technology of Windows bootkit based on MBR," *Journal of Computer Applications*, vol. 29pp. 83–85, Z1, 2009, in Chinese.