Software Consistency Checking Method for Distribution Terminal based on Chaotic Map

Yaokun WANG, Ying LIANG, Changkai SHI and Shilei GUAN
(China Electric Power Research Institute, Haidian District, Beijing, 100192 China)
Corresponding author’s e-mail: wangyaokunvip@sina.com

Abstract: Distribution terminal system software is the core carrier of its function realization and performance stability. In the process of development and operation and maintenance of distribution terminals, software versions are confusing and inconsistent with the archived versions due to technical upgrades, version management, and regulatory deficiencies. It makes some functions unable to operate normally, performance indicators cannot be met, equipment on-line rate decreased, and seriously affecting the operation and maintenance and practicality of distribution automation system. In order to strengthen the software version control of distribution automation terminal equipment, the feature information of software version was extracted by analysing the executable files of terminal software, and then the priority was divided according to the feature information. The MD5 algorithm and CRC32 checking algorithm were used to calculate the comparison factor, which then was encrypted by using pseudo random code generated by Logistic chaotic map. The model and method of software version consistency detection for distribution terminal were established.

1. Introduction
Distribution terminal is the key equipment to improve the level of distribution automation, and plays an important role in the operation and maintenance management of distribution network [1-2]. At present, there are many manufacturers of distribution terminal equipment, but the manufacturing level is uneven, and the quality of products is quite different. In the process of development and operation and maintenance of distribution terminals, the inconsistency between the actual software version and the archived version and the arbitrary modification of the software due to technical upgrade, version management, lack of supervision and other reasons make some functions unable to operate normally and performance indicators cannot be met, adversely affecting the practicality of distribution automation. According to statistics, more than 10% of the distribution terminals are offline for more than 7 days per month due to software failure. Therefore, in order to effectively ensure the practicality of distribution automation system, it is particularly important to strengthen the consistency detection and control of software version of distribution automation terminal equipment and prevent software from arbitrary modification.

At present, a more complete consistency detection method has been developed in the field of software version control abroad. Comparatively speaking, domestic is not mature enough in this field, but has also made some achievements. Digital signature, CRC verification, MD5 verification and other technologies have been used to a certain extent. However, most of the existing technologies are based on the consistency comparison model of the terminal software source code, and depend too much on the software development platform, programming language and operating system [3]. In practice, the terminal manufacturer is reluctant to disclose its source code for the sake of technical secrecy, which...
makes the consistency comparison model based on source code less practical. In addition, due to the high security and reliability requirements of the power industry, there is no special version consistency testing method for distribution terminal software.

Therefore, on the basis of meeting the above requirements, this paper designed a version consistency detection method of distribution terminal software based on chaotic encryption algorithm.

2. HASH algorithm and chaotic encryption

2.1 MD5 algorithm and CRC check

HASH algorithm, also known as hash function, is widely used in software consistency comparison. The algorithm can map any file to a fixed length hash value, that is, if the two hash values are different, their corresponding original files are different, so we can judge whether the file has been modified by comparing the hash values. MD5 message digest algorithm is the most widely used hash function, which maps any file to a 128-bit (16-byte) hash value, and the process is irreversible [4].

However, theoretical analysis shows that the hash value and the original file are not one-to-one correspondence, different original files may generate the same hash value, this situation is called collision. The probability of collision is related to the specific HASH algorithm. For example, MD5 algorithm calculates the original file and generates 128-bit hash value, so there are only $2^{128}$ different hash values, and the original file is infinite in theory, so using hash value to judge the file, inconsistent conclusions are certain, and consistent conclusions are equivocal.

In order to reduce the collision rate of MD5 algorithm, MD5+CRC 32 check is used to check the key information. In this paper, CRC checking is converted into binary system by serializing the data splicing which will be checked. CRC32 ($n=32$) checking code is calculated and saved. The data which will be checked is converted into binary system after serializing the same splicing. The 32-bit CRC checking code is then spliced at the end of the splicing, and the cyclic division is performed. Check whether the check is correct according to whether the remainder is 0.

2.2 Chaotic encryption

Chaos is a deterministic, stochastic-like process with divergence and non-periodicity, which is very sensitive to system parameters and initial values [5].

Chaotic mapping can obtain pseudo-random sequence or pseudo-random code similar to random sequence, which is called chaotic sequence. The correlation of chaotic sequences is very weak and has good white noise characteristics. The generation process of chaotic sequence is very simple. In theory, a long chaotic sequence can be obtained by iteration of chaotic mapping, and the long-term evolution result of chaotic sequence is unpredictable [6]. This makes chaotic system of great cryptographic value, and is very suitable for sequence information encryption. The application of chaotic system to encryption can simplify the encryption process [7].

Chaotic encryption is to encrypt plaintext by using chaotic sequence generated by chaotic system. The encryption process is reversible, and the plaintext signal is extracted by the same chaotic system[8]. The general process of chaotic encryption is shown in the Figure 1.

![Chaos Encryption and Decryption Process](image)

Figure 1 Chaotic Encryption and Decryption Process

Logistic mapping is a kind of classical mapping in chaotic system, and it is a dynamical system originated from population statistics[9]. The equation is:
\[ X_{n+1} = \mu X_n (1 - X_n) \]

The two parameters that affect the equation are the initial value \( X_0 \) and the system parameter \( \mu \). The results show that when the following two conditions are satisfied, the Logistic function works in a chaotic state, and the resulting sequence is nonperiodic, divergent and pseudo-random.

1. \( 3.5699456 < \mu \leq 4 \);
2. \( 0 < X_0 < 1 \).

3. Distribution terminal software version consistency model

3.1 Software feature information

Usually, the source code of the terminal software can not be obtained when testing the version of the terminal software, so it is necessary to obtain the characteristic information about the software version from the executable file of the terminal running. Select the file name, content, size, file modification time, release date and release / change description as the feature information of the software version.

Among them, the size and content of the software can directly reflect the state of the source code of the software, if the source code changed, the size and content will certainly change. The name, modification time and release date of the software reflect the archive information of the software, which is an important external feature of the software. Release / change description does not directly reflect the source code status of the software, and this feature information is only used to prompt the field acceptor of software changes and modifications, and to remind some installation and debugging considerations.

Because the consistency of terminal software is represented differently by different feature information, it can be prioritized according to the importance of feature information, which can be divided into three levels: Key information, Important information and General information. The degree of importance ranges from high to low in order of K, I and G. K-level information can directly characterize the consistency of software versions, that is, if K-level information does not match, the software has been modified; I-level information can not directly characterize the consistency of software versions, but as an important information, when I-level information does not match, alarms should be generated and confirmed by the detector; when G-level information does not match, there is no alarm, but will prompt the scene installer that G level information has changed.

3.2 Software version consistency model

3.2.1 Consistency comparison factor

Because of the large number of software version feature information and the diversity of form and content, it is not conducive to direct comparison. Therefore, HASH algorithm is considered to calculate the hash value of software feature information, and the calculated hash value is used as the comparison factor of software version detection.

In order to simplify the processing, multiple feature information of the same priority is spliced in a specific order after obtaining the corresponding hash values by corresponding algorithms. The importance level corresponding to each feature information and the encryption algorithm used are shown in the Table 1.

| Serial number | Name               | priority | encryption algorithm  |
|---------------|--------------------|----------|-----------------------|
| 1             | name               | I        | MD5                   |
| 2             | content            | K        | MD5+CRC32             |
| 3             | Size               | K        | MD5+CRC32             |
| 4             | Modification time  | I        | MD5                   |
| 5             | Release date       | I        | MD5                   |
| 6             | Release / change description | G | CRC32 |

Table 1 Priority and Encryption of Feature Information
According to the length of the output string of the encryption algorithm, MD5 outputs 16 bytes (128 bits) and CRC32 outputs 4 bytes (32 bits). Therefore, three different hash values of different lengths are obtained by calculation: K-level hash value 40 (16+4+16+4) bytes, I-level hash value 48 (16+16+16) bytes, and G-level hash value 4 bytes. Hash values are used as comparison factors in software version consistency checking. By comparing these hash values, we can judge whether the software has changed.

3.2.2 Consistency ratio factor encryption
After the distribution terminal software passes the test, the version consistency comparison model will be sent to the terminal manufacturer for comparison in site acceptance. However, it is not advisable to send the consistency comparison factor directly to the manufacturer. In order to prevent the comparison factor from being tampered illegally and the errors occurring in the transmission process, it is necessary to encrypt the generated consistency comparison factor.

Using the Logistic chaotic map, the initial value $X_0$ and the system parameter $\mu$ are determined, and the chaotic sequence $(X_1, X_2, X_3, L)$ is generated. The encrypted sequence with a length of 40 is selected from the Mth data and is marked as $M = (X_M, X_{M+1}, X_{M+2}, X_{M+39})$. The encrypted sequence with length 48 is selected as $N = (X_N, X_{N+1}, X_{N+2}, X_{N+47})$ from the Nth data, and the encrypted sequence with length 4 is selected as $L = (X_L, X_{L+1}, X_{L+2}, X_{L+3})$ from the Lth data. The five parameters of $X_0, M, N$ and $L$ are saved as Logistic encryption keys.

Each element in the three encryption sequences of $M, N$ and $L$ is represented as finite precision binary numbers. The formula is as follows:

$$a = \sum_{n=0}^{m-1} 2^{-(n+1)} b_n$$

In the formula, $a$ is a decimal encryption sequence; $b_n$ is binary data; $m$ is binary data’s length.

In this paper, the first 8-bit binary number of $b_n$ is chosen to represent the three encrypted sequences, which are transformed into 320 bit, 384 bit and 32 bit respectively.

The binary encryption sequence will be spliced and compared with the original comparison factor, and new data is obtained. The three new data sets are used as a consistency comparison model and sent to the manufacturer in two-dimensional code or some other form, which is the digital proof of the terminal software and used for comparison in the field acceptance. The specific process is shown in Figure 2.

![Figure 2: Comparison Factor Encryption Process](image)

3.3 Software version consistency comparison method
According to the process of distribution terminal detection and field acceptance, a terminal field inspection and acceptance method based on the version consistency comparison model of terminal software is designed. The flow chart is shown in Figure 3.

The on-site acceptance personnel respectively make corresponding operation according to the results of the conformance judgment. If K-level information is inconsistent, the terminal software is considered tampered with; if G-level information is inconsistent, it should be considered that the software has undergone general modification, and the field personnel should pay attention to its prompt information; if I-level information is inconsistent, it is considered that the software has undergone important modification, it should be submitted to the inspector to confirm whether the modification is legal. The I level information processing flow is shown in Figure 4.

4. Summary
Based on the analysis of software status of distribution automation terminal, the requirement of software consistency detection and control was put forward. The software feature information was extracted without knowing the source code, and the comparison factor based on MD5 algorithm and CRC32 was established according to the priority. In order to prevent illegal tampering with the comparison factor, a consistency comparison model of distribution terminal software was established by using chaotic encryption algorithm. Through analysis, the practicability of the model was verified. It is proved that the consistency detection method of terminal software designed in this paper plays an important role in improving the quality control capability of distribution terminal equipment, maintaining the consistency between the actual software version and the filed version, and preventing the illegal tampering of the software. The terminal equipment runs stably and promotes the practicability of distribution automation system.

References
[1] Cong W, Sheng Y.R, Xian G.F. (2018) Distributed power recovery method based on intelligent distribution terminal.Automation of Electric Power Systems.,42(15):77-85.
[2] Han G.Z, Xu B.Y,SuoNanJiaLe.(2012) Realization of automatic discovery technology in distribution termina.Automation of Electric Power Systems.36(18):82-85.

[3] Lv Y.J, Xu W.J, Liu Y.(2016) Design of software recording and comparison system for smart meter based on cryptography technology. Power System Technology.40(11):3604-3608.

[4] Zhang Y.Z, Zhao Yi,Tang X.B. (2008)Research on MD5 algorithm,Computer Science.07:295-297.

[5] Zhang B. (2005)Nonlinear chaos in power electronic converters and its application.Transactions of China Electrotechnical Society.12:1-6+12.

[6] Thompson J M T,Stewart H B.(1986)Nonlinear dynamic and chaos.John Wiley& Sons Ltd.

[7] Jia H.J,Yu Y.X,Wang C.S. (2001)Chaotic phenomena in power system and related research Proceedings of the CSEE.07:27-31.

[8] Qiu S.S,Chen Y.F,Wu M. (2002)Some problems of chaotic secure communication and a new scheme of chaotic encryption. Journal of South China University of Technology(Natural Science Edition). 11:75-80.

[9] Wang D.S, Cao L. (1995)Chaos, Fractal and Its Application. China University of Science and Technology Press, Hefei.