Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105
Volume **9**, Number **1**, February **2024**

Software Protection Technology on Tamper-Proof Hybrid Coded Chaotic Watermarking

Li Li*

College of Information Engineering
Chengdu Vocational & Technical College of Industry
Chengdu 610213, P.R. China
lily_better310@163.com

Di Hu

School of Science and Technology
Assumption University
Bangkok 10700, Thailand
hudi0506@126.com

*Corresponding author: Li Li
Received March 26, 2023, revised May 25, 2023, accepted September 25, 2023.

Abstract. *: Software is an essential digital product in people's daily lives, but copyright protection of software products has become a very important issue. Software watermarking can store the information of software copyright and user identity into the software in advance and stealthily, so as to protect the legal copyright of the software product owner. This paper firstly introduces in detail the generation of dynamic watermark, the process of watermark extraction and embedding, and the watermark encoding method. Secondly, based on the study of existing dynamic graph watermarking algorithms, a new software watermarking scheme based on chaos optimisation is proposed to solve the problems of poor robustness of software watermarking and difficulty of recovery after watermark destruction by introducing chaos theory. The scheme uses chaotic systems to generate chaotic sequences to split the watermark information matrix and displace the chaos. After the watermark is embedded, chaotic encryption is used to protect the entire code. It is then a hybrid encoding technique that combines the advantages of the base K-linked table encoding scheme and the PPCT encoding scheme, with the addition of tamper-proof features. When the program is run the topology must be decoded before the constants can be obtained, so when the topology structure is corrupted the whole program will not run. Simulation results show that the tamper-proof hybrid coded chaotic watermarking scheme has good stealth and robustness. Therefore, the proposed scheme is a dynamic graph coding scheme with good performance.*
Keywords: Software watermarking; Dynamic watermarking; Chaotic sequences; Hybrid coding; Robustness

1. **Introduction.** With the rapid development of computer networks and the increasing popularity of digital products, people can edit, modify and copy the original versions of digital products, including video, audio, images and software, causing the problems of intellectual property protection and information security of digital products to become more and more prominent. This has made it very difficult to protect the copyright of digital products, and it also shows that research into various technologies for protecting the copyright of digital products is urgent. In the long run, it will bring huge losses to the digital product market and even affect the normal operation of the digital product

market [1,2,3]. Therefore, how to protect the integrity, security and copyright of digital products from infringement has become a very serious problem we are facing.

The technique traditionally used to protect the copyright of digital products is encryption [4,5]. The so-called encryption technique is the step of using certain technical means to turn important data that is not wanted to be seen by others into a garbled transmission whose meaning cannot be perceived, i.e. ordinary text is combined with a string of numbers, i.e. a key, to produce an unintelligible ciphertext. This technique simply enables the transmission of secrets between the sender and receiver of the message [6,7], and once the digital product has been received and successfully decrypted by a third party, the product is in an insecure and unprotected state. A third-party attacker of this digital product can then copy and distribute it at will.

Software watermarking is a branch of digital watermarking, a cross-cutting research area of information security, cryptography, graph theory, algorithm design [8,9], and software engineering. Software watermarking technology embeds software copyright information and user identity information into the software in advance, which cannot or will be difficult to remove while maintaining the functionality of the program. This information can be extracted to detect illegal copying and piracy of software products when piracy occurs and proof of copyright ownership is required. Software watermarking technology can provide owner identification, attribution verification, copy control and piracy tracking and location. While software protected by encryption techniques can be cracked by an attack, all information about the software product is completely known to the attacker, software watermarking works by adding secret watermark information to the software [10,11], which can be extracted when needed to prove copyright. It is in this context that software watermarking has come into being. In view of the better performance of dynamic graph watermarking compared to other watermarking algorithms in all aspects, the main work of this paper is to propose a new and better performance scheme for dynamic graph software watermarking based on existing dynamic graph software watermarking coding methods. A chaotic sequence is generated using a chaotic system to partition the watermark information matrix and displace the chaos. After the watermark is embedded, chaotic encryption is used to protect the entire code. In addition, in order to better achieve the goals of high data embedding rate, concealment and good attack resistance, the two existing encoding schemes are based on and mixed with anti-tampering features to achieve better attack resistance. Finally, the experimental results are used to verify the feasibility of the proposed algorithm scheme.

1.1. **Related Work.** Software watermarking belongs to a multidisciplinary research field [12], including cryptography, algorithm design, software engineering, graph theory and programming. In recent years, research on software watermarking technology has focused on static watermarking algorithms and dynamic watermarking algorithms. Static watermark in a particular format stored in the program code, with the generation of flexible, simple identification and other characteristics, is the most commonly used watermarking algorithms, but the security is poor. Typical algorithms for this type of watermarking include: software watermarking algorithms based on image watermarking, watermarking algorithms based on basic block control flow encoding, watermarking algorithms based on software control flow graph merging, watermarking algorithms based on instruction frequency encoding, tamper-proof watermarking algorithms based on encryption functions, robust watermarking algorithms based on chaos theory dynamic graphics, watermarking algorithms based on register allocation, etc.

Dynamic watermarking makes full use of the executable features of the software itself [13], and is a hot topic of research because of its higher robustness compared to static

watermarking. In terms of watermarking characteristics, dynamic watermarking can be classified as Easter Egg watermarking, dynamic data structure watermarking, dynamic execution process watermarking, etc. Easter Egg watermarking is a watermark detector placed directly in the software code, and the watermark detector is activated by a specific input sequence. The detector, upon detecting the watermark, displays the watermark information to the user. Dynamic data structure watermarking embeds the watermark into the data structure that is dynamically generated when the software is run. When the software receives a specific input sequence, the watermark information is encoded in data such as a stack or global variable. The watermark is extracted by detecting the relevant state of the software. Dynamic execution process watermarking encodes the watermark by the sequence of execution of instructions or memory addresses in the program. The watermark is extracted by detecting the watermark information by examining the execution path diagram or data flow diagram of the program at a specific input.

Currently, dynamic watermarking algorithms using graph structure encoding are an important direction in software watermarking technology. The first dynamic graph watermarking algorithm was proposed by Collberg and Thomborson, called the CT algorithm [14], which embeds the watermark into the dynamically generated graph structure of the program, where the watermark information is hidden in the running state of the program, and the detection and extraction of the watermark must be performed in the state of program execution. Sahu et al. [15] proposed a watermark subgraph by splitting the watermarked graph. Ma et al. [16] presented a dynamic software watermarking algorithm based on multi-constant coding. Li et al. [17] proposed a chaos-based software watermarking algorithm scheme. The scheme protects the entire program code by introducing a chaotic system that does a heteroskedastic operation on the chaotic sequences and watermark information, and encodes the result as a hash into the entire code, while resisting reverse engineering attacks by embedding an anti-reverse engineering module. The above software watermarking algorithms are not perfect in terms of functionality and performance, and have the disadvantage of poor resistance to attack.

1.2. **Motivation and contribution.** The main dynamic graph watermarking algorithms in use today are the base K-linked table encoding scheme [18] and the PPCT encoding scheme [19]. Among them, the base K-linked table scheme is simple in structure, easy to implement and high in data embedding rate, but it is also more vulnerable to attacks because of its simple structure. PPCT encoding scheme is a binary tree based encoding method, which has good steganography but is not ideal in terms of data embedding rate.

The main innovations and contributions of this study are shown below:

(1) To improve the robustness of software watermarking as well as its anti-tampering performance, this work combines chaotic systems and code encryption techniques, using interleaving and mutual protection mechanisms to complete code integrity detection and protect the entire code. When the watermark is sensed to be tampered with, a tamper response mechanism is activated to terminate the program or execute the wrong control flow, preventing further damage to the software watermark by means such as reverse engineering attacks. Theoretical analysis and experiments show that this mechanism can effectively enhance the robustness of watermarking.

(2) K-link table encoding has a relatively high data embedding rate, while PPCT encoding has better resistance to attack. To better improve all aspects of the performance of the watermark encoding scheme, this work uses a hybrid encoding of the two and adds anti-tampering features to achieve better performance in terms of resistance to attack. In this way, when the program is run the topology must be decoded before the constants

can be obtained, so when the topology structure is corrupted the whole program will not run.

2. A model for software watermarking.

2.1. Definition and working mechanism of software watermarking.

Software watermarking technology is an emerging research direction in the last decade or so. Software watermarking technology is an after-the-fact verification technology [20], and its purpose is usually not to prevent software products from being pirated, but to extract specific information embedded in the software when it is pirated or illegally exploited, which is used by the copyright owner of the software to prove their copyright and thus to settle the matter through legal channels.

Let P be the original program, w be the embedded watermark information, k be the encryption key, and e be the watermark embedding algorithm, then $e(P, w, k) \to P_w$. By using the key k and the watermark embedding algorithm e, the watermark information w is embedded into the original program P, resulting in the program P_w containing the watermark information w. To ensure the correctness of the program operation, the algorithm e needs to satisfy the following constraints without changing the program function Rule [21, 22]: Let $CoI(P)$ be the set of input sequences of P and $out(P, I)$ be the output of P when the input is I. For the user, it is sufficient to observe whether the execution behaviour of the two programs P and P_w is the same. r is the watermark extraction algorithm, then $r(P_w, k) = w'$. algorithm, which extracts the watermark information w' from P_w according to the key k. Obviously, to ensure the correctness of the watermark extraction, $w = w'$ is required. The basic model of software watermarking is shown in Figure 1.

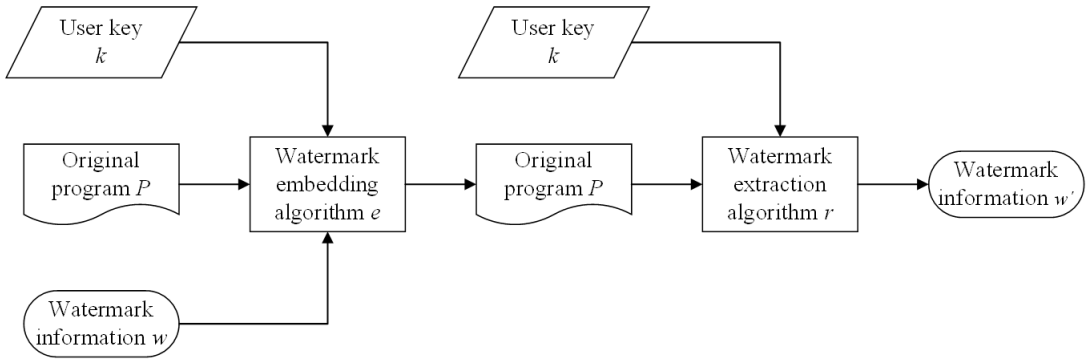

Figure 1. Basic model of software watermarking

The difference between a software watermark and a normal multimedia watermark is that the embedded watermark must not have any impact on the functionality of the software. Also the impact on the execution efficiency of the software must be as low as possible, so that the user does not perceive a significant change in the execution speed of the software with the watermarked information embedded.

When designing software watermarking algorithms, several issues need to be considered. Firstly, there is the issue of the data embedding rate, i.e. the ratio of the space taken up by the watermark to the space added to the host program in order to embed the watermarked information; secondly, the characteristics of the program format and code in which the watermark is embedded also have an impact on the characteristics of the software watermarking algorithm; and finally, the software watermark has to be designed in response to possible attack methods to better resist these attacks.

2.2. **Dynamic software watermarks.** Dynamic software watermarks (DGW) differ from static software watermarks in that instead of embedding the watermark information in the static text of the program, the watermark is embedded in the dynamic execution environment of the program. Dynamic software watermarks are characterised by the need for a specific input sequence as a key, where the program is run into a certain state and the process of extracting the watermark information is extracted from the execution state of the program.

The process of embedding and extracting the dynamic graph watermark is as follows [23]:

(1) Enter the watermark information W to be embedded.

(2) Represent the watermark information W as a watermark number (i.e. an integer) and transform it into some topology G. The topology G may be one of the structures such as a base K-link table, a permutation graph, an enumeration tree or a PPCT.

(3) Partition the graph G into several subgraphs $G_0, G_1, \ldots, G_m$.

(4) Encode each subgraph G_i and convert it to the corresponding program code C_i.

(5) Embed $C_0, C_1, \ldots, C_m$ in the program mark position according to the predefined input sequence $i_0, i_1, \ldots, i_n$, to complete the embedding of the watermark.

(6) When extracting the watermark, the program containing the watermark is executed. After entering the correct predefined sequence $i_0, i_1, \ldots, i_n$, the program calls the corresponding watermark construction code $C_0, C_1, \ldots, C_m$, and dynamically generates the graph G_i in memory. When the last graph G_m is generated, the entire topological graph G is restored according to the watermark segmentation algorithm, and then it is into the corresponding original watermark information W.

2.3. **Dynamic image watermark encoding methods.**

(1) Base link table encoding.

The base K-linked table encoding consists of k nodes, including a head node and a circular chain table consisting of k nodes. The base 5 chain table code is shown in Figure 2 and is represented as follows: $438 = 3 \times 5^3 + 2 \times 5^2 + 2 \times 5^1 + 2 \times 5^0$.

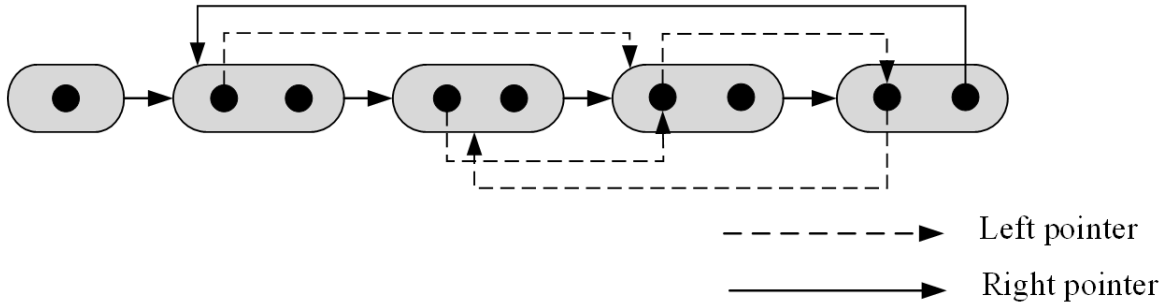

Figure 2. Base 5 chain table code

In base K-linked table encoding, the head node contains only one pointer to the first node in the circular chain table. Each node in the circular link table has two pointers, left and right, where the left pointer is used for encoding the watermark information. The right pointer is used to point to the next node in the circular chain table. The right pointer of the last node points to the first node in the circular chain table. Thus, for encoding a base K-linked table containing k nodes, the number of watermarks that can be represented ranges from 0 to $k^{k-1} - 1$. The encoding function for a base K-linked table

is

$$Index = \sum_{n=1}^{k-1} a_n k^{n-1} \tag{1}$$

Where a_n is the multiplication factor. If the left pointer of the node is NULL, $a_n = 0$; if the left pointer of the node points to itself, $a_n = 1$.

(2) PPCT code.

The PPCT coding structure is an improvement from a binary tree.The PPCT is a special binary tree that adds an Origin node [24,25] to the binary tree and makes the pointer to the Origin node point to the root node.The PPCT adds a right pointer to the Origin node and the leaf nodes to the special binary tree structure so that all leaf nodes of the binary tree and the generating nodes form a circular chain table, as shown in Figure 3. The PPCT structure has good resistance to attack. Next we analyse the

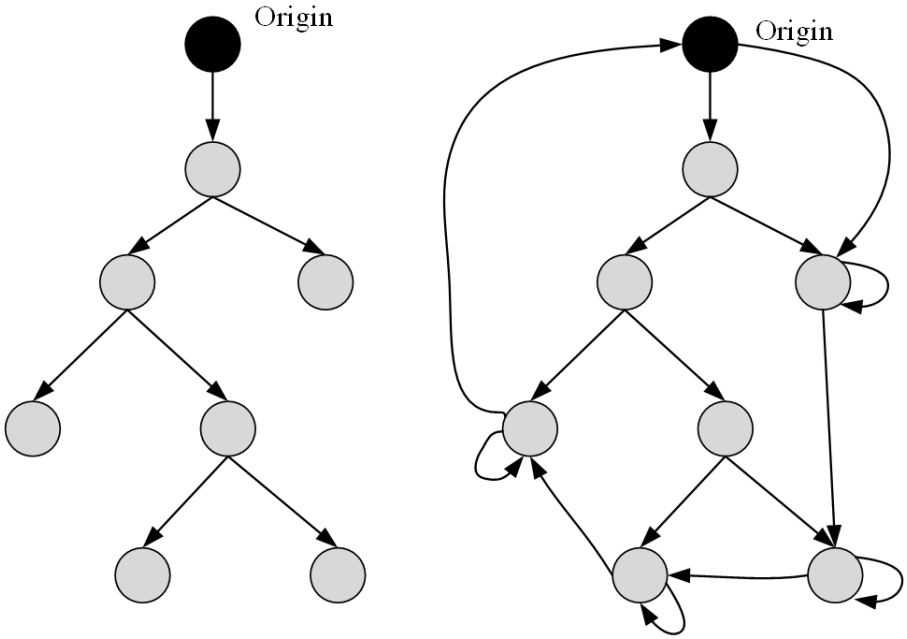

Figure 3. Special binomial trees and PPCT structures

process of transforming a PPCT structure into a watermark number N. Assume that the watermark number N represented by the PPCT structure $T = \text{int}(T)$, T.left denotes the left subtree of T, T.right denotes the right subtree of T, and L and R denote the number of leaf nodes in the left and right subtrees of T respectively.

$$\text{int}(T) = 0, if|T| = 1 \tag{2}$$

$$\text{int}(T) = \text{int}(T.left) \times C(R) + \text{int}(T.right) + \min\_\text{int}(L, R), if|T > 1| \tag{3}$$

where $\min\_\text{int}(L, R)$ is the smallest integer.

$$\text{min\_int}(L, R) = \min \text{int}(L-1, R+1)+C(L-1)\times C(R+1) = \sum_{i=1}^{L-1} (C(L-i)\times C(R+i)) \tag{4}$$

Topologically, the PPCT structure has the characteristics of both a binary tree and a single circular chain table, and uses pointers for structure generation. Since the exact value of the pointer is different each time the program is run, it increases the difficulty for an attacker to break the watermark. Also, according to the characteristics of the PPCT structure, for a PPCT structure with m leaf nodes, as soon as any node in the tree is

found, the generating node can be found in m-1 steps along its left pointer, thus traversing the whole tree. Secondly, when an attacker modifies the pointers of some nodes in the PPCT structure, it is even possible to effectively recover the PPCT structure based on its generation rules, which has a strong error correction capability.

2.4. **Types of Software Watermarking Attacks.** We currently classify software watermarking attacks into four main categories: addition attacks, removal attacks, distortion attacks and complicity attacks.

The most effective form of attack among the above-mentioned attacks is the hold semantic transformation attack in the deformation attack. The so-called hold semantic transformation attack uses obfuscation techniques to replace or otherwise change variables in a program, where the program runs unchanged but the internal structure of the program has changed dramatically and the extraction of the watermark becomes relatively problematic. Each of these hold semantic transformations brings a degree of obfuscation to the program, and in practice these hold semantic transformation algorithms are not usually used alone, but rather a combination of multiple semantic hold transformations are used together so that they can provide sufficient obfuscation to the program.

3. Chaos optimization based tamper-proof hybrid coded watermarking scheme.

3.1. **Chaos theory.** Chaos is a description of the emergence of a non-linear dynamical system with similar random uncertain output, disorder containing order, which is indecomposable, regular and unpredictable.

Logistic mapping is a chaotic dynamical system [26], which has a very complex dynamical behavior and has a wide range of applications in fields such as communication and cryptography, where the dynamics of a one-dimensional function can be expressed as:

$$x_{k+1} = \mu x_k \left(1 - x_k\right) \tag{5}$$

where the chaotic domain is $(0, 1)$ and μ is the branching parameter.

The Logistic mapping works in the chaotic state when $3.5699456 < \mu < 4$. When given the initial value x_0, the sequence $\{x_k; k = 0, 1, 2, 3, \ldots\}$ generated by the one-dimensional chaotic function under the Logistic mapping is acyclic, non-convergent, and closely related to x_0.

$$S_n(k) = R_n\left(x_k\right) = \begin{cases} 0, x_k \in \bigcup\limits_{d=0}^{2^{n-1}-1} I_{2d}^n \\ 1, x_k \in \bigcup\limits_{d=0}^{2^{n-1}-1} I_{2d+1}^n \end{cases} \tag{6}$$

where n is any positive integer and $I_0^n, I_1^n, \ldots, I_{2n}^n$ are $2n$ consecutive equal intervals. The chaotic sequence $\{x_k; k = 0, 1, 2, 3, \ldots\}$ is transformed by calling $S_n(k)$ into a binary output sequence.

3.2. **A software watermark splitting and storage scheme based on chaos optimization.** On the basis of the original CT algorithm, the split-storage watermark is formed by matrix partitioning of the watermark information, and the split-storage watermark is encrypted by using chaotic dislocation CS.

The encrypted split-storage watermark is then represented using DPPCT and the Hash-processed watermark information is populated in the Info field of DPPCT. Finally, the sensitive code segment of the program is encrypted using Chaos Encryption. The watermark embedding process is as follows:

(1) For the watermark information W, perform matrix partitioning to form a sub-storage watermark $(w_1, w_2, w_3, \ldots, w_m)$ and transform it into matrix $\mathbf{A}$;

(2) Using chaotic dislocation CS, dislocation encryption of matrix $\mathbf{A}$ to form matrix $\mathbf{A}'$;

(3) Pre-processing of program source code, marking the location of watermark embedding and program sensitive code segment CSB, adding Hash functions, encryption and decryption functions;

(4) The sub-stored watermark information w_i' ($1 \leq i \leq m$) for each row of matrix $\mathbf{A}'$ is encoded in the PPCT topology map and embedded in the location specified by the program. In the running state of the program by the user input sequence $\{I_1, I_2, I_3, \ldots, I_m\}$;

(5) Compile to generate the target program P_w';

(6) Apply the chaotic encryption algorithm CE to encrypt P_w'.

The watermark extraction process is the inverse process of watermark embedding. Firstly, the target program P_w' is chaotic decrypted CE^{-1} by user input key, then the matrix $\mathbf{A}'$ is extracted and the matrix $\mathbf{A}$ is obtained from the inverse of the chaos SR^{-1}. Finally, the watermark information W is restored by the matrix partitioning algorithm.

The proposed algorithm converts the watermark information W into a matrix $\mathbf{A}$. First the watermark information W is converted into a binary message. The watermark information W is partitioned according to the length of the binary information L_w and the length of each segment is L_{w_i+1}. The processed watermark information is noted as $W = (w_1, w_2, w_3, \ldots, w_m)$. Then construct a matrix $\mathbf{A}(A_m \times A_n)$, where the number of watermark branches $A_m = m$ and the length of each segment $A_n = \lfloor L_w \rfloor + 1$. The segmented watermark information is placed in each row of the matrix in order.

For example, the watermark information $W = 29$ is converted to binary as 1101. According to the above algorithm $A_m = 2$ and $A_n = 3$, then $\mathbf{A} = \begin{bmatrix} 111 \\ 010 \end{bmatrix}$

The proposed algorithm can generate $2m \times n$ chaotic sequence values and can displace and encrypt any $m \times n$ matrixs. The permutation encryption process is as follows:

(1) Generation of a sequence C containing $2m \times n$ chaotic sequence values by a chaotic system;

(2) Matrix partitioning of the chaotic sequence C to obtain $2m \times n$ matrices C_1, C_2;

(3) The matrices A and C_1 are dissociated by rows to form a new matrix T_1, i.e. $T_1 = A \oplus C_1$;

(4) Transpose T_1 to form a new matrix T_2;

(5) The matrix T_2 and C_2 are dissociated by rows to form the new matrix T_3, i.e. $T_3 = T_2 \oplus C_2$;

(6) Transpose T_3 to form a new matrix T_4.

The above is an iterative permutation of matrix A. The resulting matrix T_4 is the chaotic permutation of matrix A'. To get a better result, the above process can be iterated several times.

The inverse process is: first, transpose the matrix T_4 to obtain the matrix T_5, then transpose T_5 with C_2 to obtain the matrix T_6, then transpose T_6 to obtain the matrix T_7. Finally, transpose T_7 with C_1 to obtain the matrix T_8. The matrix T_8 is the original watermark information matrix A.

3.3. **Improved tamper-proof hybrid encoding watermarking scheme.** Based on the analysis above we can know that the base k-linked table encoding has a relatively high data embedding rate, while the PPCT encoding has better resistance to attacks. In order to better improve the performance of the watermarking scheme in all aspects, we use a hybrid encoding method of both and add tamper-proof features to achieve better performance in terms of resistance to attack.

We know that the main advantage of encoding base K-linked table is the high data embedding rate, but the structure is too simple and can be easily detected and compromised. ppct is the more commonly used topology graph, which is resistant to attack and concealment, but does not have a high data embedding rate. Therefore, we have combined the two, thus inheriting the advantages of both. The advantages of PPCT are guaranteed on the basis of its node right pointer pointing to encode it, thus implementing the idea of a base K-linked table encoding scheme. The constants are then extracted from the software program and represented as integers. Finally, the encoding scheme is transformed into a similar graph topology. The constants in the program are replaced with these graph topologies, thus creating a pseudo watermark to confuse the attacker. Therefore, when the pseudo-watermark information is tampered with or removed from the program, the constants are also corrupted and cannot be extracted properly, and the entire program will not run. The structure of the tamper-proof hybrid coding scheme is shown in Figure 4.

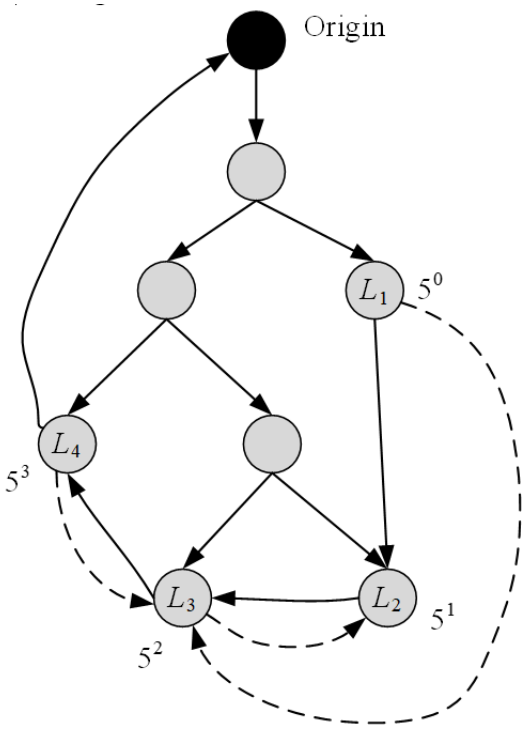

Figure 4. Tamper-proof hybrid code

If there are n leaf nodes, the Origin node in the topology diagram points to a node of power 0, the next node pointed to by the left pointer of that node is of power 1, the next two nodes pointed to are of power 2, and so on until the left pointer points to the generating node of power n-1.

$$c_i = \begin{cases} 0 & j = i \\ j - i + 1 & j \geq i \\ n - |(j - i)| + 1 & j < i \end{cases} \tag{7}$$

The value of ci is represented by the information of the right pointer of its leaf node, which ranges from 0 to n+1.

Based on the characteristics of the hybrid encoding structure, we form a sufficiently large watermark library. We can choose any one of these as our topology to encode the

watermark information, which is very effective in resisting complicity attacks. The steps for implementing the tamper-proof hybrid encoding scheme are

(1) Select the watermarked number N, which can be decomposed into two large prime numbers;

(2) Calculate the number of leaf nodes required for the topological graph structure;

(3) Select a hybrid coding structure based on the number of leaf nodes;

(4) Selecting constants from the source program and encoding them into the PPCT topology structure;

(5) Embedding the coded information into the software program.

Based on the principle that large numbers are hard to decompose, we choose the product of two large prime numbers as the watermark information we want to embed. We choose the *BigInteger* type, which can represent a larger range of numbers, to represent the watermark number and use the *probablePrime()* method of the *BigInteger* class to generate the large prime number.

$$Public\ static\ BigInteger\ probablePrime\ (int\ bitLen,\ Random\ rnd) \tag{8}$$

In general, the numbers returned by this method are all prime numbers. The *bitLen* parameter determines the number of bits in the binary corresponding to the prime number returned, and the rnd parameter is used to test the random number type variable returned. Next, a 32-bit prime number is obtained.

$$Random\ rand = new\ Random\ ()\ ; \mathrm{Re}turn\ BigInteger.ProbablePrime(32,\ rand); \tag{9}$$

The prime numbers returned when the function is executed are randomly generated, and the product of the two randomly generated large prime numbers is used as the watermarked number to be embedded. This makes it impossible for an attacker to fully decompose the two large prime numbers within the limited time or capability available, even if they were able to find the exact watermarked numbers to be embedded. Thus, the copyright of the software is very well protected by this method.

When encoding constants create an encoding function *encode(int i)* in the program, use this function to implement the conversion from constants to graph structures, i.e. to form a pseudo watermark graph. When the program runs, we need to convert the pseudo-watermarked graph into a constant again, this process is the decoding of constants, it is the inverse of the constant encoding process. The *decode* function *decode(CG[i])* can be created to reduce the pseudo watermark structure to a constant. The pseudo code for the watermarking procedure is shown in Table 1.

3.4. **Chaotic Encryption CE.** This work combines chaotic sequences with encryption algorithms to construct chaotic encryption schemes with variable length cipher and interleaved protection mechanisms.

Using hashes of non-sensitive code segments [27], the watermark information and sensitive code segments are protected from attackers tampering with the watermarking system, and the encryption process is shown in Figure 5. During the execution of the program, the Decrypt function is called to decrypt CSBi when the ciphertext CSBi code block is encountered. Then, the plaintext code block CSBi is executed. The encryption process uses the chaotic sequence qi as an argument with a variable-length key, which can effectively make it more difficult for an attacker to crack [28], and the hash value of CIBi as an argument, which can effectively prevent the whole program from being tampered with by an attacker. After the program has finished running, the encryption module is called again to complete the protection of the sensitive code segment.

4. **Simulation experiments and performance tests.**

Table 1. Pseudocode for watermarking procedures.

The process of a tamper-proof hybrid coding scheme

Input: Watermark information W, user input sequence $\{I_1, I_2, I_3, ..., I_m\}$, user input key k.
Ouput: Mixed Wm; //The watermark map is a mixed coding structure.
public method () {
int C[1], C[2];
Wm = buildWm (); //Embed watermark.
C[1]=100;
C[2] = 200;
Print(C[1]+C[2]);
//The pseudo-code for the watermarking procedure to perform constant encoding.
Mixed Wm, CG[1], CG[2];
public method () {
int C[1], C[2];
CG[1] = encode(1); //Embed the 1st pseudo watermark.
CG[2] = encode(2); //Embed the 2nd pseudo watermark.
Wm = buildWm(2);
C[1] = decode(CG[1]); // Extract the constant from the 1st pseudo watermark.
C[2] = decode(CG[2]); //Extract constants from the 2nd pseudo watermark.
...
Print(C[1] + C[2]);
}

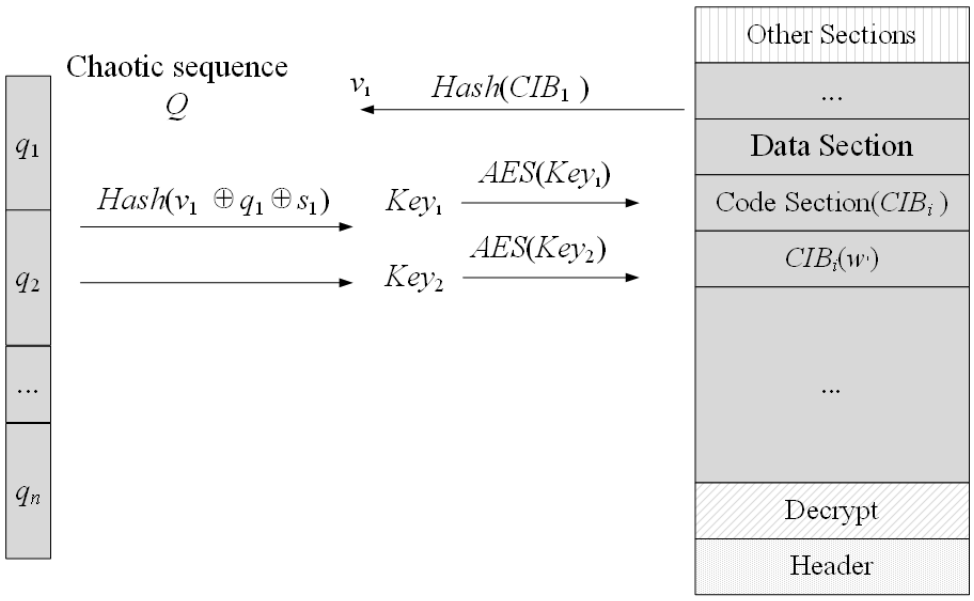

Figure 5. Chaotic Encryption Process

4.1. **Experimental platform.** The tamper-proof hybrid coding scheme is validated through simulation experiments. The system platform for the algorithm used for software watermarking is the SandMark platform.

The SandMark platform is an open source system developed by Christian Collberg and others at the University of Arizona, which allows the embedding and extraction of

watermarks into programs and the testing of programs against obfuscation and tampering attacks. The main features of the SandMark platform are: software watermarking algorithms, code obfuscation, code optimisation, bytecode viewing, static data statistics, bytecode comparison and static code analysis functions. With these features, we can use the SandMark platform to embed watermarks on programs and simulate various attacks to test the performance of watermarking algorithms. In addition, as the platform is plug-in structured, we can add the watermarking algorithms proposed in this work to the platform.

4.2. **Performance testing and analysis.** This work uses the SandMark software watermarking test platform, invokes the dynamic graph watermark CT algorithm in it, completes the generation, embedding and extraction of watermark data, and completes the integrity detection and anti-tampering of the watermark with the help of FlexHEX, OllyDbg, XJad and other tools.

The watermark data generation module mainly converts the watermark information W entered by the user into a watermark number and then encodes the watermark number into a graph structure. The tamper-proof hybrid coded watermarking scheme proposed in this work is compared and analysed with the PPCT coding scheme and the base K-linked table scheme.

4.2.1. *Robustness analysis.* The topology of the hybrid encoded watermarking scheme is constantly changing in the stack and has multiple graph structures, making it difficult to find the exact location of the watermark.

Also due to the changed graph topology, the resistance of the watermark to pattern matching and complicity attacks is enhanced. Secondly, with the newly added Pointer and Info domains, the PPCT structure is characterised by a bi-directional circular chain table and makes every node except the Origin node contain watermark information, further improving robustness and security. A comparison of the attack resistance of the hybrid coded watermarking scheme with other coding schemes is shown in Table 2.

Table 2. Comparison of attack resistance analysis of 3 coding schemes

Coding schemes	**Adding attacks, clipping attacks**	**Twisted Attack**	**Fault tolerance**
Base K-linked table	weak	weak	General
PPCT Code	Strong	weak	Strong
Mixed coding	Strong	Stronger	Stronger

It can be seen that chaotic dislocation and chaotic encryption of the watermark by a chaotic system yields good random uncertainty outputability and autocorrelation properties, thus effectively resisting differential and linear attacks on known plaintexts. If the two-dimensional chaotic parameter μ is cracked, 2 correlated value pairs of that chaos are required. For a chaotic system with n chaotic sequence values, the cracking probability is n^{-2}. If a code segment in the program is modified, this will cause the decryption to fail and the program to terminate.

4.2.2. *Data rate analysis.* The hybrid coding structure combines the features of both base K-linked table coding and PPCT coding.

The hybrid coding structure has $2n$ nodes which can be coded in the range of 0 to $n^{n-1} - 1$. The data rates of the three methods of base K, PPCT and hybrid coding are compared when the number of nodes is given and the results are shown in Table 3.

Table 3. Comparison of data rate analysis for different coding methods

Number of nodes	Base K-linked table	PPCT Code	Mixed coding
n	$n^{n-1}-1$	$2C_{n-2}^{n/2-1}/n$	$(n/2)^{n/2-1}-1$
10	1.00 x 109	1.40 x 101	6.25 x 102
20	5.24 x 1024	4.86 x 103	1.00 x 109
30	1.78 x 1083	1.29 x 1012	3.55 x 1033

4.2.3. *Performance overload analysis.* The embedding of the watermark will definitely have an impact on the execution efficiency and performance of the program, mainly in the form of space occupation and time increase.

Experiments were conducted using the SandMark platform to process the TIT.jar program, embedding different numbers of watermarks and analysing the impact on the original program size and running time with different numbers of watermarks, the results are shown in Figure 6 and Figure 7.

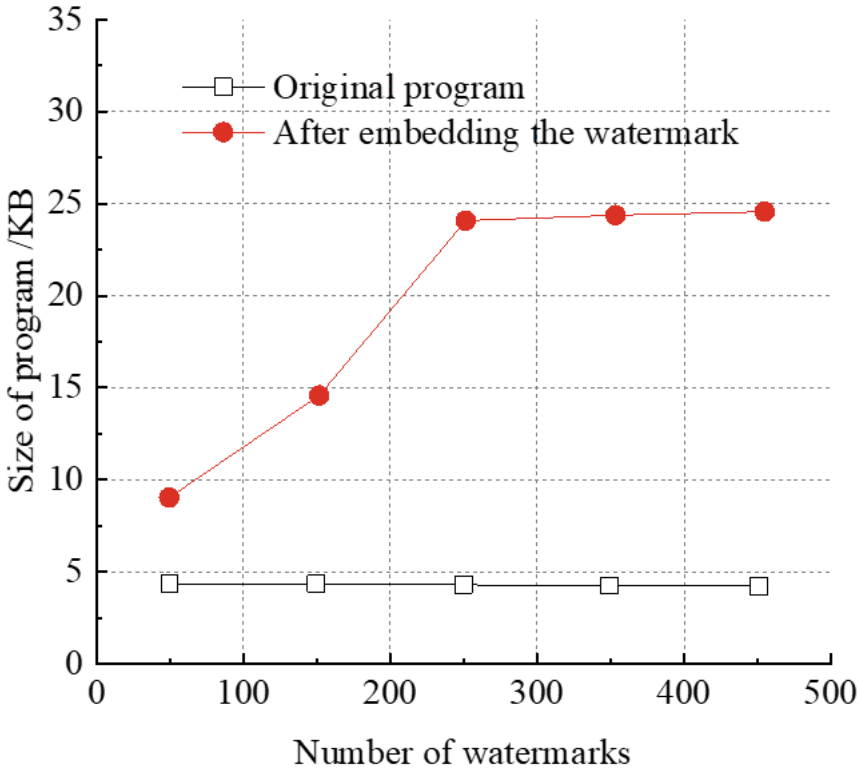

Figure 6. Program size change

It can be seen that the program size increases slowly after watermark splitting. This is due to the matrix partitioning of the watermark information, which increases the number of watermarks in the sub-store linearly, and the use of mixed coding with a higher data rate, which reduces the program load. It can be seen that the program runtime is not significantly affected by the embedding of the watermark. This is due to the fact that the embedded watermark code does not participate in the main functional modules of the program and only chaotic encryption is applied to sensitive code segments of the program, which does not significantly increase the program's overall running time.

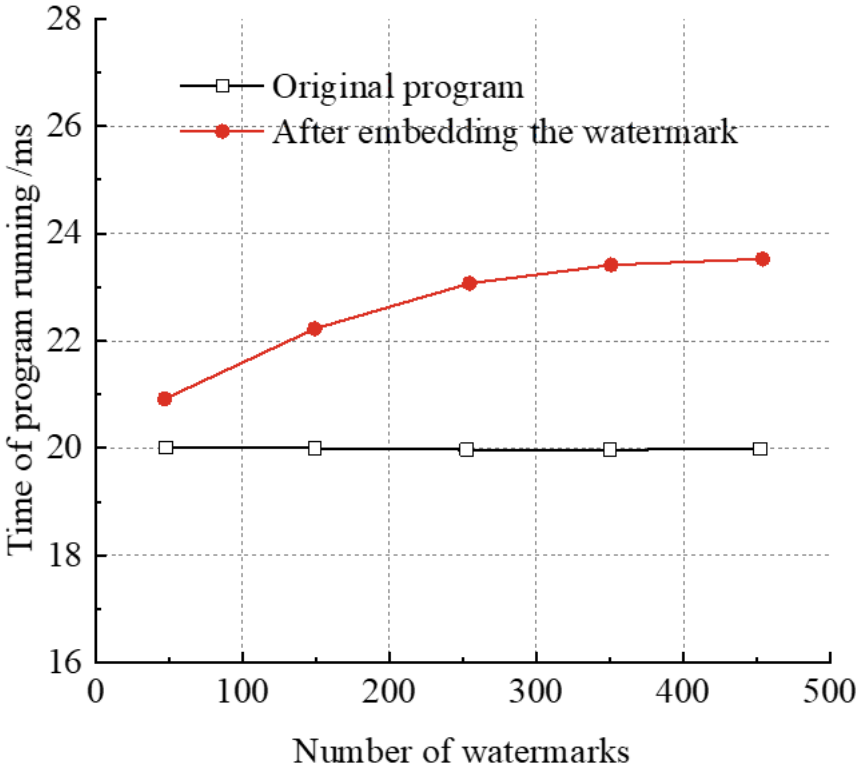

Figure 7. Program runtime change

5. **Conclusion.** This work proposes to see a chaos-optimized software watermarking scheme, which improves the stealth of software watermarking through chaos dislocation and watermark splitting, and constructs a hybrid coded PPCT topology graph structure to improve the robustness and attack resistance of watermarking. At the same time, it combines chaotic encryption to protect the watermark information and all the code to prevent the software and watermark from being tampered with. Theoretical analysis and experiments show that this scheme can significantly improve the robustness and error correction capability of the watermark. In addition, this work takes a hybrid coding approach to embedding the watermark and incorporates anti-tampering features to obtain better resistance to attacks. The proposed dynamic graph software watermarking scheme is simulated on the SandMark platform. The results show that the new scheme is also significantly better and more resistant to attack in terms of stealth and resistance to attack. Dynamic graph watermarking is difficult to detect by an attacker during program execution due to the dynamically changing graph structure. However, for an experienced attacker, it is easier to find the code that builds the graph by decompiling the program and observing the changes in the stack. Therefore, subsequent research will focus on analysing how to improve the stealthiness of the watermarked code.

REFERENCES

[1] T.-Y. Wu, Q. Meng, Y.-C. Chen, S. Kumari, and C.-M. Chen, "Toward a secure smart-home IoT access control scheme based on home registration approach," *Mathematics*, vol. 11, no. 9, 2123, 2023.

[2] T.-Y. Wu, F.-F. Kong, Q. Meng, S. Kumari, and C.-M. Chen, "Rotating Behind Security: An enhanced authentication protocol for IoT-enabled devices in distributed cloud computing architecture," *EURASIP Journal on Wireless Communications and Networking*, vol. 2023, 36, 2023.

[3] X. Guo, W. Jiang, Q. Zhang, and K. Wang, "Digital Protection Technology of Cultural Heritage Based on ArcGIS Geographic Information Technology Algorithm," *Security and Communication Networks*, vol. 2022, pp. 1-10, 2022.
[4] X. Li, C. Liao, and Y. Xie, "Digital Piracy, Creative Productivity, and Customer Care Effort: Evidence from the Digital Publishing Industry," *Marketing Science*, vol. 40, no. 4, pp. 685-707, 2021.
[5] T.-Y. Wu, Q. Meng, L. Yang, S. Kumari, and M. P. Nia, "Amassing the Security: An Enhanced Authentication and Key Agreement Protocol for Remote Surgery in Healthcare Environment," *Computer Modeling in Engineering and Sciences*, vol. 134, no.1, pp. 317-341, 2023.
[6] S. Hilbolling, H. Berends, F. Deken, and P. Tuertscher, "Complementors as connectors: managing open innovation around digital product platforms," *R&D Management*, vol. 50, no. 1, pp. 18-30, 2019.
[7] C.-M. Chen, Y. Hao, and T.-Y. Wu, "Discussion of 'ultra Super Fast Authentication Protocol for Electric Vehicle Charging Using Extended Chaotic Maps'," *IEEE Transactions on Industry Applications*, vol. 59, no. 2, pp. 2091-2092, 2023.
[8] L. Wessel, A. Baiyere, R. Ologeanu-Taddei, J. Cha, and T. Blegind Jensen, "Unpacking the Difference Between Digital Transformation and IT-Enabled Organizational Transformation," *Journal of the Association for Information Systems*, vol. 22, no. 1, pp. 102-129, 2021.
[9] C.-M. Chen, L.-L. Xu, K.-H. Wang, S. Liu, and T.-Y. Wu, "Cryptanalysis and Improvements on Three-party-authenticated Key Agreement Protocols Based on Chaotic Maps," *Journal of Internet Technology*, vol. 19, no. 3, pp. 679-687, 2018.
[10] S. Namasudra, G. C. Deka, P. Johri, M. Hosseinpour, and A. H. Gandomi, "The Revolution of Blockchain: State-of-the-Art and Research Challenges," *Archives of Computational Methods in Engineering*, vol. 28, pp. 1497-1515, 2020.
[11] C.-M. Chen, W. Fang, S. Liu, T.-Y. Wu, J.-S. Pan, and K.-H. Wang, "Improvement on a Chaotic Map-based Mutual Anonymous Authentication Protocol," *Journal of Information Science & Engineering*, vol. 34, no. 2, pp. 371-390, 2018.
[12] J. Ma, J. Chen, and G. Wu, "Robust Watermarking via Multidomain Transform Over Wireless Channel: Design and Experimental Validation," *IEEE Access*, vol. 10, pp. 92284-92293, 2022.
[13] E. Farri and P. Ayubi, "A robust digital video watermarking based on CT-SVD domain and chaotic DNA sequences for copyright protection," *Journal of Ambient Intelligence and Humanized Computing*, vol. 3, pp. 68-85, 2022.
[14] C. S. Collberg, C. Thomborson, and G. M. Townsend, "Dynamic graph-based software fingerprinting," *ACM Transactions on Programming Languages and Systems*, vol. 29, no. 6, 35, 2007.
[15] A. K. Sahu, K. Umachandran, V. D. Biradar, and O. Comfort, "A Study on Content Tampering in Multimedia Watermarking," *SN Computer Science*, vol. 4, no. 3, 222, 2023.
[16] H. Ma, C. Jia, S. Li, W. Zheng, and D. Wu, "Xmark: Dynamic Software Watermarking Using Collatz Conjecture," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 11, pp. 2859-2874, 2019.
[17] Z. Li, H. Zhang, X. Liu, C. Wang, and X. Wang, "Blind and safety-enhanced dual watermarking algorithm with chaotic system encryption based on RHFM and DWT-DCT," *Digital Signal Processing*, vol. 115, 103062, 2021.
[18] L. Zeng, W. Ren, Y. Chen, and M. Lei, "LMDGW: a novel matrix based dynamic graph watermark," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 1, pp. 295-304, 2017.
[19] T. A. Grotzer, A. M. Kamarainen, M. S. Tutwiler, S. Metcalf, and C. Dede, "Learning to Reason about Ecosystems Dynamics over Time: The Challenges of an Event-Based Causal Focus," *BioScience*, vol. 63, no. 4, pp. 288-296, 2013.
[20] S. Che and Y. Wang, "A Software Watermarking Based on PE File with Tamper-proof Function," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 12, no. 2, pp. 1012-1021, 2014.
[21] A. Mpanti, S. D. Nikolopoulos, and L. Palios, "Strong watermark numbers encoded as reducible permutation graphs against edge modification attacks," *Journal of Computer Security*, vol. 12, pp. 1-22, 2022.
[22] D. Mata-Mendoza, M. Cedillo-Hernandez, F. Garcia-Ugalde, and A. Cedillo-Hernandez, "Secured telemedicine of medical imaging based on dual robust watermarking," *The Visual Computer*, vol. 38, no. 6, pp. 2073-2090, 2021.
[23] M. Harahap, J. R. Malau, T. N. Simangungsong, D. Winata, and D. Hadyanto, "Digital Image Copyright Protection with Spatial Domain Public Image Watermarking Scheme," *Journal of Computer Networks, Architecture and High Performance Computing*, vol. 4, no. 1, pp. 69-78, 2022.

[24] E. Elbasi, N. Mostafa, and E. Cina, "Robust, Secure and Semi-Blind Watermarking Technique Using Flexible Scaling Factor in Block-Based Wavelet Algorithm," *Electronics*, vol. 11, no. 22, 3680, 2022.

[25] L. van Iersel, R. Janssen, M. Jones, Y. Murakami, and N. Zeh, "A Practical Fixed-Parameter Algorithm for Constructing Tree-Child Networks from Multiple Binary Trees," *Algorithmica*, vol. 84, no. 4, pp. 917-960, 2022.

[26] M. Wang, S. Chen, and J. Jing, "Chaotic shadows of black holes: a short review," *Communications in Theoretical Physics*, vol. 74, no. 9, 097401, 2022.

[27] B. Ramadevi and K. Bingi, "Chaotic Time Series Forecasting Approaches Using Machine Learning Techniques: A Review," *Symmetry*, vol. 14, no. 5, 955, 2022.

[28] A. H. Bukhari, M. A. Z. Raja, N. Rafiq, M. Shoaib, A. K. Kiani, and C.-M. Shu, "Design of intelligent computing networks for nonlinear chaotic fractional Rossler system," *Chaos, Solitons & Fractals*, vol. 157, 111985, 2022.