Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/080647

The Research of Multi-point Function Opaque Predicates Obfuscation Algorithm

YANG Yubo[1,*], *FAN Wenqing*[2], *HUANG Wei*[2], *XU Guoai*[1] *and YANG Yixian*[1,3]

[1] Information Security Center, Beijing University of Posts and Telecommunications,100876,Beijing, China
[2] School of Computer Science, Communication University of China, 100024, Beijing, China
[3] School of Information Engineering, Beijing Institute of Graphic Communication, 102600, Beijing, China

Received: 17 Nov. 2013, Revised: 15 Feb. 2014, Accepted: 16 Feb. 2014
Published online: 1 Nov. 2014

Abstract: On the algorithm of code obfuscation, opaque predicate is used to confuse the judgment of the program branches to achieve complex control flow statement. Currently, there are problems of obfuscation on the opaque predicate. The first problem is the isolation. the relation between opaque predicate is relatively isolated, once an opaque predicates Boolean value is obtained, the branch flow will always be identified; Second is the reversible, an attacker can determine the opaque predicate the Boolean value by analyzing reversible slicing attacks. In this paper, we proposed multi-point function opaque predicate obfuscation algorithm, using multi-function features makes opaque predicates interdependent and avoid reversible slicing attacks. Experimental data demonstrates that the obfuscated program performance and security has been significantly improved.

Keywords: Code Obfuscation, Complicated Control Flow, Opaque Predicate, Point Function

1 Introduction

The innovation and development of computer science provide the necessary protection for the global information technology, computer software system brings convenience for the customers while its security is also got attention gradually. As the specificity of the software industry, its source code, the private information within it and the core algorithm are easily cracked and replication. This caused software protection of intellectual property huge losses.

Code obfuscation technology, which is a key technology of software security, transforms the program's source code and execution of the internal structure logic without changing the semantics of the original program. By using this way, it avoids third-party security vulnerabilities without the aid of other additional safety methods, On the other hand, it enhance the overall security of the program without saving the hidden keys such as the key information.

In 1993 Fred Cohen proposed the idea of a semantic transformation[1], which is considered first proposed concept of code obfuscation. Then Christian Collberg summarized and classified the code obfuscation techniques[2], and made an accurate definition about the code obfuscation. Converting source program P to P' by confusion algorithm T, meanwhile P and P' are equivalent logically in semantics. This conversion complying with two principles, (1) If the source program P fails or stops abnormally, the program P' is not necessarily terminated; (2) If the source program P terminates normally, the program P' must be terminated and has the same output as the program P.

In the paper[2], the authors proposed three criteria to judge the merits of the obfuscation algorithm: potency, resilience and cost. Assuming O the specified obfuscation algorithm, $P \xrightarrow{O} P'$ means obfuscation algorithm O used for the source program P is converted to the obfuscated program P'.

Definition 1(Potency): Suppose $O_{pot}(P)$ is the definition of the obfuscation potency of program P, $C(P)$ is the complexity of the program P, $C(P')$ is the complexity of the obfuscated program P', Then the formula is as follows:

$$O_{pot}(P) \stackrel{def}{=\!=} \left| \frac{C(P') - C(P)}{C(P)} \right| \quad (1)$$

* Corresponding author e-mail: satiago.yang@gmail.com

And $C(P') > C(P)$,$O_{pot}(P) > 0$.

Definition 2(Resilience): Suppose $O_{res}(P)$ is the definition of the obfuscation resilience of program P, O_{PE} represents the development cost of the anti-obfuscation tool for algorithm O, O_{DE} indicates the cost of reconstruction the program obfuscated by the algorithm O. Then the formula is as follows:

$$O_{res}(P) \stackrel{def}{=\!=} Res(O_{DE}, O_{PE}) \tag{2}$$

Definition 3(Cost): Suppose $O_{cost}(P)$ is the definition of the obfuscation cost of the program P. According to $O_{cost}(P')$, the cost of executive obfuscated program P', there are four cases : $dearcostlycheap$ and $free$, Then the formula is as follows:

$$O_{cost}(P) \stackrel{def}{=\!=} \begin{cases} dear\ , O_{cost}(P') = O(e^p), p > 1 \\ costly\ , O_{cost}(P') = O(n^p), p > 1 \\ cheap\ , O_{cost}(P') = O(n) \\ free\ , O_{cost}(P') = O(1) \end{cases} \tag{3}$$

And n is the number of program execution statement.

Complicated control flow is a relatively mature and critical technologies in code obfuscation, one of the most famous technology is flat control flow algorithm chenxification[3]by C. Wang. The program through the switch statement code blocks segmentation and reassembly to achieve the control flow of obfuscation. The algorithm of obfuscation[4,5]insert redundant control flow into control flow graph of the function, increasing the difficulty of rebuilding program control flow. The obfuscation algorithm of unconditional jump[6]through the functional similarities between assembly instructions *call* and *jmp* to obfuscate the jumps address and increase the analysis difficulty of disassembly tools.

The use of opaque predicate is critical components of the complicated control flow of the algorithm of obfuscation. Because many of the internal procedures and branching processes execution path will pass on certain variable or condition judgment to determine the next step towards the control flow, by controlling these key points in the flow of obfuscation, replacing it by opaque predicate, the adversary is difficult to infer its Boolean value from the expression. In this way, under the premise of adding the opaque predicate, the segmentation and reassembly of program code block can be more effective confuse attackers to increase the difficulty of reverse analysis.

Based on the important of opaque predicate in the algorithm of complicated control flow obfuscation, in this paper, we obfuscate the opaque predicate with the help of multi-function property. It solves the current insufficient of the opaque predicate obfuscation which are isolation and reversible. The experimental tests for the proposed algorithm based on the obfuscation criteria definition 1,2,3. Experimental results proved that the cost of the obfuscated program only shows a linear growth, but its resilience obfuscation got a significant increase.

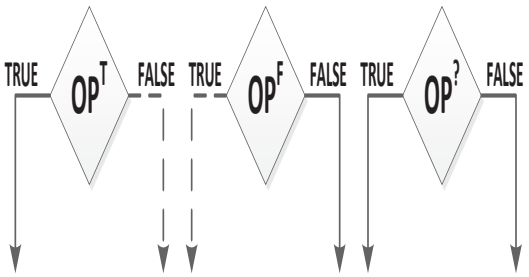

Fig. 1: Three types of opaque boolean value

2 Preliminaries

First, the opaque predicates and multi-point function definition are proposed.

Assume that Boolean expression $E(x)$ when the input is x in the program P, y is the internal compare threshold, the Boolean expressions is the following formula:

$$BOOL_{E(x)} \stackrel{def}{=\!=} \begin{cases} 0\ , E(x) = y \\ 1\ , otherwise \end{cases} \tag{4}$$

It can lead to the definition of opaque predicates.

Definition 4(Opaque predicate): Assuming $OP(x)$ is the opaque predicate expression, T_{OP} is the opaque predicate internal obfuscation transform, T_T is the true condition of the internal obfuscation transformation, T_F is the false condition of the internal obfuscation transformation , $T_?$ is the uncertainly condition of the internal obfuscation transformation, the formula is as follows:

$$BOOL_{OP(x)} \stackrel{def}{=\!=} \begin{cases} 0, T_{OP}(E(x)) = T_F(y) \\ 1, T_{OP}(E(x)) = T_T(y) \\ \{0,1\}, T_{OP}(E(x)) = T_?(y) \end{cases} \tag{5}$$

And the complexity $C[T_{OP}(E(x))] > C[E(x)]$.

The opaque predicates can be divided into three types based on the Boolean result, As shown in Fig.1:(1)If the result of an opaque predicate is always true, then labeled OP^T;(2)If the result of an opaque predicate is always false, then labeled OP^F;(3)If the result of an opaque predicate is uncertain, then labeled $OP^?$.

The dotted arrows in Fig.1 indicate the execution path program wont run. In these three types, the proposed algorithm is focused on the boolean result uncertain opaque predicate, with one-way and multi-point property of multi-point function, it can effectively obfuscate the internal logic of opaque predicates, and according to the actual situation, to determine the Boolean complete the selection of path.

Point function can be divided into two types of functions which are single point function and multi-point

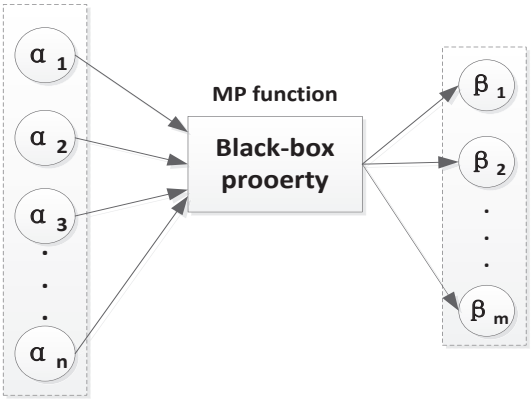

Fig. 2: Multi-function black-box properties and astringent properties

function. Single-point function will have the correct output only when the input data is equivalent to a particular data point, while Multi-function will have the correct output when the input data is equivalent to the specific data point.

First, assume single-point function expressed as SP, correspondence relationship between the input and output is $\alpha \xrightarrow{SP} \beta$, then the formula:

$$SP(x) \xlongequal{def} \begin{cases} \beta \ , x = \alpha \\ \perp \ , otherwise \end{cases} \tag{6}$$

Thus leads to the definition of Multi-point function.

Definition 5(Multi-point function): Assume multi -point function expressed as MP, the set of input variables is α^n, the set of output is $\beta^{m(n)}$, m which is number of elements of the output set β, is the function of the number n of input variables of element, and $n \geq m(n)$. When $\alpha_i \in \alpha^n$,$\beta_j \in \beta^{m(n)}$, then the formula:

$$MP(x) \xlongequal{def} \begin{cases} \beta_j \ , x = \alpha_i \\ \perp \ , otherwise \end{cases} \tag{7}$$

From Fig.2, Multi-function relationship between input and output is not one correspondence but a trend of convergence, which enhances the multi-function black-box property, making the adversary increase the difficulty of its reverse. The implement in the multi-point function can use the hash function to enhance the internal obfuscation.

3 Related work

3.1 The opaque predicate obfuscation

The research of the opaque predicate obfuscation focused on several aspects, first is achieving the opaque predicate by pointer alias[5], the obfuscation algorithm through the following steps to achieve:

(1)In the way of dynamically allocated, program code constructs a set of graphs $G = \{G_1, G_2, G_3, \ldots\}$;

(2)There is a pointer set $P = \{p_1, p_2, p_3, \ldots\}$ in the program, and set $R = \{R_1, R_2, R_3, \ldots\}$ with a fixed relationships between the set of graphs G and the set of pointer P;

(3)With the relation set R, in the maintenance of the unchanged relationships of set R, modified set G and set P achieves an opaque predicates specific relationship;

In addition to achieve opaque predicate[3] through the array alias can be well hidden the control flow. The basic idea is that a specific location in an array saves the data which has in common mathematically in these locations, such as $X \equiv 5 mod 3$, and the other positions filled junk data. As long as the relationship between data in the array stable, the value of the data can be updated in real time. And data in the array can be used to generate opaque predicate, because of the random of number, increased the difficulty of reverse.

Because of the cross-semantic of the concurrent executive program, it is difficult to reverse analysis. Suppose there are n concurrent executions in a program statement, then there will be $n!$ possibility in the execution order, the analysis difficulty of multi-thread is the condition to generate the opaque predicate. The basic steps of obfuscation algorithm[7] are as follows.

(1) In the program P generated a global one-way circle list C, C list structure is fixed, with n nodes;

(2) In the program P two pointers A_1 and A_2 are constructed, and initializes it to point to the same node in the list C;

(3) In the program P two pointers B_1 and B_2 are constructed, and initializes it to point to the different node in the list C;

(4) Create a new thread T_1, the pointer A_1, A_2 are updated by thread T_1 asynchronously, when each update operation is completed, the pointer A_1, A_2 move to the next node according to the order in the list C;

(5) Create a new thread T_2, the pointer B_1B_2 are updated by thread T_2 asynchronously, when each update operation is completed, the pointer B_1, B_2 move to the next node according to the order in the list C;

Opaque predicate is constructed shown in Fig.3, $(A_1 = A_2)$ result is constantly true, $(B_1 = B_2)$ result is constantly false; the adversary does not know the fixed relationship of pointer in the thread, determining the value of the opaque predicate is very difficult.

There are two problems about the research of opaque predicate obfuscation. (1)Regardless the way how to achieve opaque predicate, the relationship between the opaque predicates is less or isolated, it provides the adversary with the possibility of crack individually. (2)The opaque predicate black-box property are not strong enough, it provides the possibility of reversing the opaque predicate internal logic for the adversary.

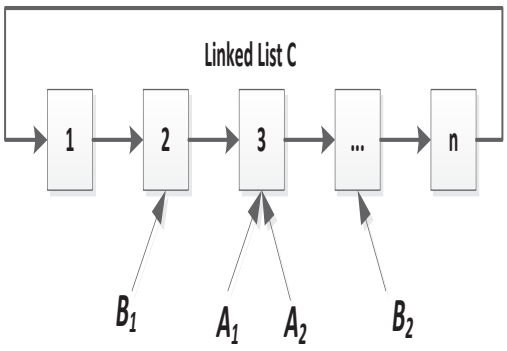

Fig. 3: The structures and pointer status of list C

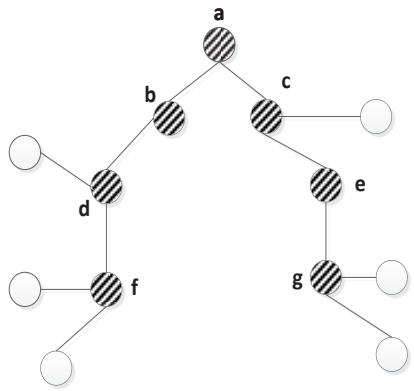

Fig. 4: A graph traversal scene of access control

3.2 The point function obfuscation

The research of the point function obfuscation is an open problem, many studies show its proved that point function having the positive effects on the obfuscation algorithm. It is first proposed the obfuscation definition of single-point and multi-point function in the Random Oracle R in paper[8], then it was used in two different obfuscating situations.

First is to achieve the access control obfuscation by using multi-point functions. Assuming Fig.4 is a graph traversal scene of access control. Before moving from one node to the next node, it is necessary to verify the access authority of the node. In order to prevent access violation of the adversary, verification process must be a black-box property, so it is necessary to use multi-point function to do obfuscation.

Second is to achieve the regular expression obfuscation by using the multi-point functions. The regular expression can represent the finite string, such as $\wedge w(a|b|c)d\$$, it can be represented as a collection of strings $\{wad, wbd, wcd\}$. It is effective to hide the structure of the regular expression by forge string status which is not existed through multi-point functions obfuscation.

Paper[9] proposed the algorithm to obfuscate the database access using multi-function, access the confused database through multi-point function, but the access efficiency decreased significantly.

The effective research of point function[10] based on the mathematical model of obfuscation[11]. From the angle of mathematic, the author proved that single-point function enhances the obfuscation effects, also discussed about the future work of multi-point function research.

Based on the research status of opaque predicates and point function, the author proposed the algorithm based on multi-point function opaque predicates obfuscation($MPOP$): (1)To achieve opaque predicate properties interdependence through multi-point function so that the adversary can not obtain the control flow of the program by analyzing separate opaque predicates; (2)Based on combination of multi-point functions and *hash* functions, enhance opaque predicates black box property and the adversarys difficulty of reversing the obfuscation program.

4 The algorithm of MPOP

First, assume that obfuscation model O based on the algorithm $MPOP$, $B \in B_n$ is branch statement can be confused in the model, and n represents the size of the input data. $O_{mpop}(B)$ represents the obfuscation of branch statement opaque predicates in the model, Then the obfuscation model needs to meet the following three conditions:

(1)functionality

Assuming there is a negligible function $\alpha(n)$ to all $B \in \{B_n\}$, $O_{mpop}(B)$ and B are function similar, the formula is expressed as:

$$func[O_{mpop}(B)] - func(B) \leq \alpha(n) \tag{8}$$

(2) polynomial slowdown

Assuming there is to all a polynomial p to all $B \in \{B_n\}$, The maximum value of cost within the scope of the polynomial, the formula is expressed as:

$$cost[O_{mpop}(B)] \leq p(|B|) \tag{9}$$

(3) black-box property

Obfuscation model O scale of opaque predicates is m, arbitrary function $\varepsilon(m) = 1/m^{O(1)}$. Assuming there is another obfuscation model O', and the confusion model on a scale of polynomial function $s(m, 1/)$, when m is large enough, to all $B \in \{B_n\}$, R is a model for anti-obfuscating. the formula is expressed as:

$$\left|Pr\left[R\left(O_{mpop}(B)\right) = 1\right] - Pr\left[R\left(O'(B)\right) = 1\right]\right| \leq \varepsilon(m) \tag{10}$$

MP function
Input: $\{\alpha\} = \{\{\alpha_B\}, \{\beta\}\}$
Begin:
  if$(hash(T(\{\alpha\}) \vDash C_A))$
    β_A=*RelatedValue*$(\{\alpha\})$;
    return $hash(\beta_A)$;
  if$(hash(T(\{\alpha\}) \vDash C_B))$
    β_B=*RelatedValue*$(\{\alpha\})$;
    return $hash(\beta_B)$;
  if$(hash(T(\{\alpha\}) \vDash C_C))$
    β_C=*RelatedValue*$(\{\alpha\})$;
    CreateBranchPath(β_C);
    return $hash(\beta_C)$;
End
Output: $hash(\{\beta\})$

Fig. 5: Pseudo-code of MP function algorithm

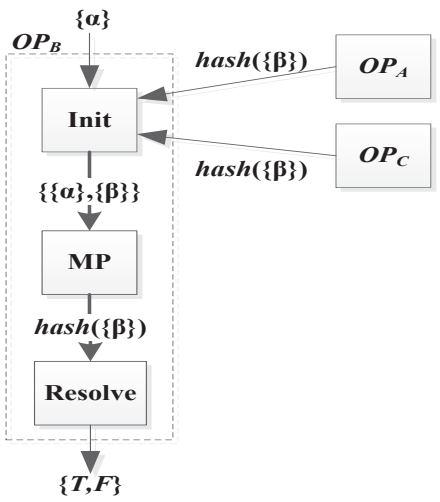

Fig. 6: Execution processes of *MPOP* algorithm

Based on the obfuscation model *O*, assuming there are three opaque predicates OP_A, OP_B, OP_C and they execution order is $OP_A \rightarrow OP_B \rightarrow OP_C$. And three execution states are different, OP_A has been executed, OP_B is ready to execute, OP_C is not execute. In order to obfuscate the multi-point function, need to define the internal logic of multi-point function first, Let $\{\alpha\}$ is the input data set of multi-point function of OP_B.

According to the characteristics of the input data of the multi-point function, There are three internal function execution conditions represent as C_A, C_B, C_C. T represents the logical transformation of the input elements of the set $\{\alpha\}$, the set $\{\alpha\} = \{\{\alpha_B\}, \{\beta\}\}$, β is related value, this variable is used to associate other opaque predicates such as OP_A,OP_C, making opaque predicates interdependence.

Condition *A*: if $T(\{\alpha\})$ meet C_A, and generates related value β_A, β_A value is related to the set $\{\alpha\}$,but there is no boolean property.

Condition *B*: if $T(\{\alpha\})$ meet C_B, the branch conditional path is executed according to the real boolean value, and generates related value β_B, The value of β_B is related with the set $\{\alpha\}$ and the current boolean value.

Condition *C*: if $T(\{\alpha\})$ meet C_C, forge false branch execution path, and execute the path which is random selected, and generates related value β_C, The value of β_C is related with the set $\{\alpha\}$ and the boolean value actually executed.

Multi-function algorithm pseudo-code in Fig.5, all critical data were processed using fully minimal hash function[12],which enhance the security of obfuscation.

With the property of multi-point function, according to the input data set $\{\alpha\}$ and related value set $\{\beta\}$ to choose the branch execution in variety of ways. Since the function output is not clear-text boolean value, but as the hidden information contained in the related value, and processed by hash function, which also ensures the security of information transmission. opaque predicates OP_B as example for the execute process of *MPOP* algorithm.

MPOP algorithm shown in Fig.6 is divided into three phases:

(1)Initialize variables

When the input is a set $\{\alpha\}$, which is a parameter, obtaining hash value of the related value opaque predicate OP_A, OP_C. On the one hand making related values of other opaque predicates as the input parameters of multi-point function in OP_B, on the other hand making the opaque predicates related to each other to determine the other opaque predicates not be replaced or modified.

(2)Executing MP function

When the input parameter set for the $\{\alpha\}$ and $\{\beta\}$, through the logical transformation of parameter set in the multi-point function, according to the results, to select the corresponding conditional execution, and output the hash values of related values.

(3)Analyzing related value

Analyzing the input related value $\{\beta\}$, to determine the boolean value of branch structure when the opaque predicate is performed.

Based on the *MPOP* algorithm, the opaque predicate achieves the interdependence between opaque predicates with the related value. If the opaque predicate replaced or modified, Program will be abnormal and cannot execute correctly for obtaining the invalid related value. This method effectively avoids the isolation of opaque predicate. We use the black-box properties of the multi-point function and algorithm of a one-way *hash* in *MPOP* algorithm. If the adversary wants to reverse

opaque predicate correctly, then the adversary has to crack the hash algorithm and the internal logic of multi-function which would pay a heavy price. So through this way, it will enhance the anti-reverse of opaque predicates.

5 Experiment

5.1 Technology realization

In order to implement the *MPOP* algorithm in this paper, we use a code obfuscation tool:LOCO[13], which can control the program flow of confusion automatically or manually, with interactive visual interface and cross-platform features.

First, analyze the binary file of sample program and generate the corresponding control flow graph. the main reason of using the binary file as input is to eliminate the differences of code language and system platform, improve the accuracy of LOCO analysis and ensure the uniformity and comparability of experimental results.

Second, the target of the proposed algorithm is to confuse the branch structure of sample program, so the control flow graph generated by other uninteresting code only need to be simplified representation. This method can improve the efficiency of LOCO analysis. Because of visual interface of LOCO, This effect can easily be realized.

Finally, after generating a control flow graph, we write a program to automatically insert confusion node named $INSERT-OB$ in order to achieve the program's automatic confusion. Use the program to quickly traverse the entire control flow graph, locate the branch structure which needs to be confused and insert *MPOP* to the branch structure.

By LOCO analyzing and obtained the control flow graph of sample programs, locating the branch node and obfuscating it with the *MPOP* algorithm. For comparison test results, we take the obfuscation algorithm $AA(ArrayAlias)$[3] currently and widely used to confusing the sample programs.

5.2 Sample program collection

Experimental test sample programs are from the benchmark programs of $SPECint-2006$ benchmark suite[14], $SPECint-2006$ benchmark includes 12 test programs, we take five of them as test sample set, the selected sample programs are listed in Table 1.

Experimental test platform is 2.9GHz Intel Core2 Duo CPU, RAM 4GB, operating system platform is the Ubuntu 11 version, the compiler is gcc version 3.4, the optimization level O3. IDA Pro tool[15] is used to reverse the test program, the confusion result evaluate the effect of the algorithm.

Table 1: Experimental sample program

Program	Language	Compiler	Platform
401.bzip2	C	gcc	ubuntu
429.mcf	C	gcc	ubuntu
445.gobmk	C	gcc	ubuntu
456.hmmer	C	gcc	ubuntu
458.sjeng	C	gcc	ubuntu

5.3 Evaluation Metrics

Refer to the previous three confusion algorithm evaluate criteria definition 1,2,3: potency, resilience and cost, we elicit a concept: confusion factor[16], used to measure the results of confusion.

CF_{pot} defines the confusion factor to evaluate the confusion potency, CFP_{before} defines the total number of instructions before the confusion, CFP_{after} defines the total number of instructions proper resolved by disassembler. Then, the formula as follows:

$$CF_{pot} = \left| \frac{CFP_{before} - CFP_{after}}{CFP_{after}} \right| \tag{11}$$

CF_{res} defines the confusion factor to evaluate confusion resilience, CFR_{before} defines the total number of the opaque predicates, CFR_{after} defines the total number of opaque predicates proper resolved by disassembler. Then, the formula as follows:

$$CF_{res} = \left| \frac{CFR_{before} - CFR_{after}}{CFR_{after}} \right| \tag{12}$$

CF_{cost} defines the confusion factor to measure the confusion cost, the comparison sample programs are the original program, the program based on the obfuscation algorithm *AA* and the program based on the obfuscation algorithm *MPOP*. The running time can be tested by the same test sample program group, CF_{cost} can be graded based on the growing trend of the running time.

$$CF_{cost} = \begin{cases} dear\ , CF_{cost} = O(e^p), p > 1 \\ costly\ , CF_{cost} = O(n^p), p > 1 \\ cheap\ , CF_{cost} = O(n) \\ free\ , CF_{cost} = O(1) \end{cases} \tag{13}$$

5.4 Performance

In the test experiments, the original program were confused for opaque predicates by the obfuscation

Table 2: Confusion potency comparison

Program	*AA* algorithm	*MPOP* algorithm
bzip2	75.45%	85.23%
mcf	78.56%	89.47%
gobmk	80.24%	87.16%
hmmer	68.47%	78.35%
sjeng	73.15%	87.69%
Average	75.17%	85.58%

Table 3: Confusion resilience comparison

Program	*AA* algorithm	*MPOP* algorithm
bzip2	55.68%	75.36%
mcf	63.12%	78.45%
gobmk	70.24%	90.12%
hmmer	65.35%	80.56%
sjeng	72.84%	89.32%
Average	65.45%	82.76%

algorithm of *AA* and the obfuscation algorithm of *MPOP* proposed in this paper, the experimental results of two algorithms were compared.

First of all,the comparison of the confusion potency is shown by the confusion sample program test data, the results are listed in Table 2.

Second, the comparison of the confusion resilience is shown by the confusion sample program test data, the results are listed in Table 3.

Finally, the comparison of the confusion cost is shown by the confusion sample program test data, the results are listed in Fig.7.

The experimental data shows in Table 2,3 means that comparing to the opaque predicates obfuscation algorithm of *AA*, the disassembly error rate of the *MPOP* algorithm increased by nearly 20% (from 65.45% to 82.76%). The experimental data in Fig.7 shows that, *MPOP* algorithm is based on the black box property of multi-point function and it does not need as many read and write operations as the *AA* algorithm, so the *MPOP* algorithm shows the linear running time growth. However, with the test data

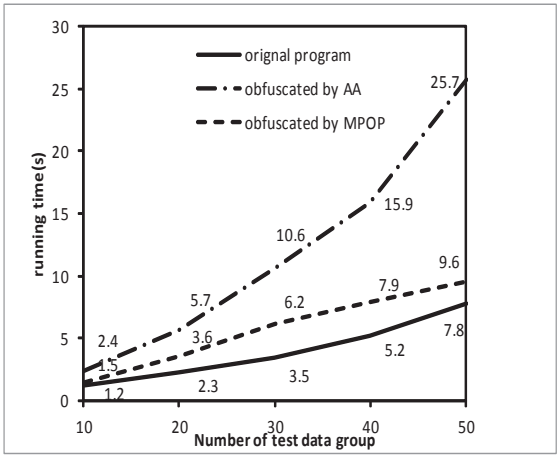

Fig. 7: Degree distributions of the above 8 samples

increase, the running time of the *AA* algorithm shows the exponential growth but the significantly decrease of efficiency.

6 Conclusions

This paper presents a new confusion algorithm of opaque predicates, the MPOP algorithm use the advantage of the specificity of multi-point function and improves the effect of opaque predicate confusion. This algorithm is not only a good solution to the two defects of current confusion opaque predicate: isolation and reversible, but also enhance the security of confusion program. The comparison of confused sample program only shows a linear growth, but its resilience obfuscation got a significant increase.

Acknowledgement

This work is supported by National Natural Science Foundation of China (No.61121061), the Specialized Research Fund for the Doctoral Program of Higher Education (20120005110017) and National Key Technology R&D Program (2012BAH06B02). The authors are grateful to the anonymous referee for a careful checking of the details and for helpful comments that improved this paper.

References

[1] F. B. Cohen, Operating system protection through program evolution, Comput. Secur., vol. 12, no. 6, pp. 565584, 1993.
[2] C. Collberg, C. Thomborson, and D. Low, A taxonomy of obfuscating transformations, Department of Computer Science, The University of Auckland, New Zealand, 1997.

[3] C. Wang, J. Davidson, J. Hill, and J. Knight, Protection of software-based survivability mechanisms, in Dependable Systems and Networks, 2001. DSN 2001. International Conference on, 2001, pp. 193202.
[4] C. Collberg, C. Thomborson, and D. Low, Manufacturing cheap, resilient, and stealthy opaque constructs, in Proceedings of the 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, 1998, pp. 184196.
[5] C. S. Collberg, C. D. Thomborson, and D. W. K. Low, Obfuscation techniques for enhancing software security, 6,668,325,Dec-2003.
[6] C. Linn and S. Debray, Obfuscation of executable code to improve resistance to static disassembly, in Proceedings of the 10th ACM conference on Computer and communications security, 2003, pp. 290299.
[7] J. Nagra and C. Thomborson, Threading software watermarks, in Information Hiding, 2005, pp. 208223.
[8] B. Lynn, M. Prabhakaran, and A. Sahai, Positive results and techniques for obfuscation, in Advances in Cryptology-EUROCRYPT 2004, 2004, pp. 2039.
[9] A. Narayanan and V. Shmatikov, Obfuscated databases and group privacy, in Proceedings of the 12th ACM conference on Computer and communications security, 2005, pp. 102111.
[10] H. Wee, On obfuscating point functions, in Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, 2005, pp. 523532.
[11] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang, On the (im) possibility of obfuscating programs, in Advances in CryptologyCRYPTO 2001, 2001, pp. 118.
[12] M. L. Fredman, J. Komls, and E. Szemerdi, Storing a sparse table with 0 (1) worst case access time, J. Acm Jacm, vol. 31, no. 3, pp. 538544, 1984.
[13] M. Madou, L. Van Put, and K. De Bosschere, Loco: An interactive code (de) obfuscation tool, in Proceedings of the 2006 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation, 2006, pp. 140144.
[14] [Online]. Available: http://en.wikipedia.org/wiki/SPECint.
[15] Liege and Belgium, IDA Pro. [Online]. Available: http://www.datarescue.com/idabase/.
[16] I. V. Popov, S. K. Debray, and G. R. Andrews, Binary obfuscation using signals, in USENIX Security Symposium, 2007, pp. 275290.

YANG Yubo, born in 1986, is currently a PhD candidate in Information Security Center, Beijing University of Posts and Telecommunications , China. He received his bachelor degree from North China University of Technology, China, in 2011. His research interests include information security, code obfuscation.

FAN Wenqing, born in 1983, is currently an instructor in School of Computer Science, Communication University of China. He received his PhD degree from Beijing University of Posts and Telecommunications, China, in 2010. His research interests include information security, program analysis.

HUANG Wei, born in 1983, is currently an instructor in School of Computer Science, Communication University of China. He received his PhD degree from Beijing University of Posts and Telecommunications, China, in 2010. His research interests include information security.

XU Guoai, is currently a professor in School of Beijing University of Posts and Telecommunications, China. His research interests include information security.

Yang Yixian, is currently a professor in School of Beijing University of Posts and Telecommunications, China. His research interests include information security.