Improvement of the Algorithm to Evaluate Security Strength of Share Assignment on Communication Network in SSS

Masaki Tanise[1] and Masahiro Hayashi[2]
[1,2] Major of Information, Graduate School of Integrative Science and Engineering, Tokyo City University,
1-28-1 Tamazutsumi, Setagaya-ku, Tokyo 158-8557, Japan
E-mail: [1]g2281442@tcu.ac.jp, [2]mhaya@tcu.ac.jp

***Abstract*— We propose a new algorithm for the share assignment problem, which evaluates how the security strength varies depending on where shares are stored on a communication network in the secret sharing scheme. For a long time, the secret sharing scheme was studied only in terms of how to make shares. However, a recent study indicated that the security strength depends on where shares are stored on the communications network after they have been made and proposed a method for evaluating the security strength of each allocation of shares. However, the algorithm of the proposed method is inefficient. Here, we propose an improved algorithm based on a binary tree search. Numerical experiments show that the presented method surely reduces the computer usage time compared with the original method.**

Keywords—communication networks, security, domain, secret sharing scheme

I. Introduction

The traditional approach to improving security is to encrypt data by using a secret key. However, studies such as refs. [1]-[9] claim this method is inconvenient because key management is not so easy and encryption is not always effective; for instance, a hacker could steal the hardware containing the data and spend time offline decrypting it.

Against such threats, researches such as refs. [1]-[9] show that the secret sharing scheme (SSS) can be used to secure data. This technique splits data (called the *secret*) into pieces (called *shares*) and spreads them to people called *participants* so that a hacker cannot recover the secret even if he can aggregate a certain number of shares. It does not need to manage keys and offers high security even if some participant's hardware is stolen.

Research on secret sharing started in 1979 [1][2], and numerous researches appeared after refs. [1][2], including [3]-[9]. Ref. [8] found that the strategy of assignment has a big impact on the security strength of this method, as explained below.

Today's cloud computing systems store important data on communications networks. If the participants in the secret sharing scheme are nearby, e.g., in the same DNS domain, the security strength is not so high. On the other hand, if the participants are in different DNS domains, the security strength is much higher because a hacker would have to aggregate shares from different domains.

Ref. [8] also proposed a method to evaluate the security strength. It gave a model representing a network by a graph with domains representing DNS domains, DNS zones, private networks, or other groups of nodes wherein a single person or a single section is consistently responsible for the security of the nodes in each group.

However, the evaluation algorithm of ref. [8] still has a problem in that its computation time seriously increases as the number of domains increases.

Here, we propose a new algorithm to reduce the computation time below that of the algorithm in ref. [8]; it is based on the idea of a binary tree search. Some numerical examples show the effectiveness of our proposal.

II. Preliminary

A graph is a mathematical object consisting of nodes and links. A link is defined as any pair of nodes. An example of a graph is illustrated in Fig. 1. The circles indicate nodes and the lines indicate links.

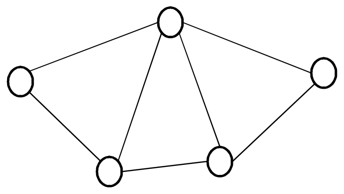

Fig. 1. Example of a graph.

A graph is denoted by $G = (V, E)$, where V is the set of nodes and E is the set of links. If a node $v \in V$ is an end node of a link e, we say that 'link e connects to v' and 'node v connects to e'. If node v_x and v_y are at either end of the same link e, then e is expressed by $e = (v_x, v_y)$. If v is deleted from the graph, the link connected to it is also deleted.

Suppose we have the following sequence of nodes and links between nodes v_{x1} and v_{xu}.

$$v_{x1} - (v_{x1}, v_{x2}) - v_{x2} - (v_{x2}, v_{x3}) - \dots - (v_{xu-1}, v_{xu}) - v_{xu}$$

Here, we say that 'v_{x1} and v_{xu} are connected'. We call the above sequence a 'path from v_{x1} to v_{xu}'.

We call v_{x1} the 'start node' and v_{xu} the 'last node'. If we do not care which nodes are the start or last, we use the word 'path' for the above sequence.

For any sets α and β, $\alpha - \beta = \{x \mid x \in \alpha, x \notin \beta\}$.
For example, $\{1, 2, 3, 5\} - \{2, 5, 7\} = \{1, 4\}$.

$P(\ )$ denotes the probability of occurrence of an event in $(\ )$.

III. Previous Researches

A. Threshold Secret Sharing Scheme

In this paper, we focus on threshold secret sharing, which is a typical SSS [1][2][7]. This technique splits the secret into n shares, and recovery of the secret is impossible unless k ($\leq n$) shares are aggregated.

B. Problem of Share Assignment

The security strength of SSS is affected not only by k and n, but also the places to which shares are assigned in the communications network. If k shares are in the same computer, the security strength is very weak. If they are in different personal computers or different DNS domains, then the security level becomes higher.

A simple example is illustrated in Fig. 2. Here, when $n = k = 2$, readers will agree that the situation of 'one share in Domain 1 and the other is Domain 2' is more secure than if both shares were in the same domain.

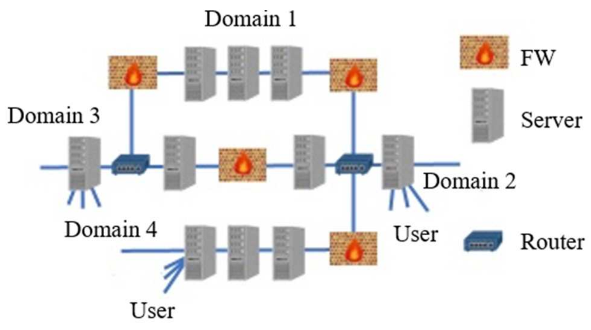

Fig. 2. Example network.

Ref. [8] first pointed out this problem and gives its mathematical formulation together with its solution.

C. Key Idea of the Mathematical Formulation

Ref. [8] pointed out that the PDCA (Plan Do Check, and Action) cycle is useful for tackling the problem. (Note that ref. [8] explained its scheme for the case of $k = n$ but it can be easily extended to the cases of $k < n$)

In the PDCA cycle, we first enumerate the feasible plans of the structure of the communications network, the places to which shares are to be assigned, and the cost constraint. Second, we evaluate the security strength of each plan. Third, we find which plan works the best by analyzing the results of the evaluations. Fourth, we implement the best plan and feedback our implementation experience to the first step.

The difficulty here is that many factors affect the security strength of SSS, in particular, the number of paths from the hacker to the participants and the security strength of the hardware and software on those paths.

Below, we explain how to overcome this difficulty by imagining an example not related to communications networks.

Countermeasures against viral epidemics are executed by nations, local governments, companies, and other organizations. When confronted with an outbreak, these organizations will try to close the areas under their control. If their efforts are successful, a pandemic will be avoided. Human society consists of many sub-societies within certain areas, and each sub-society has the responsibility and right to control their areas. Supposing that we call a sub-society a 'domain', the strength to resist the virus depends on the strength of each domain and the structure of the domains. Fig. 3 is a conceptual view of this situation.

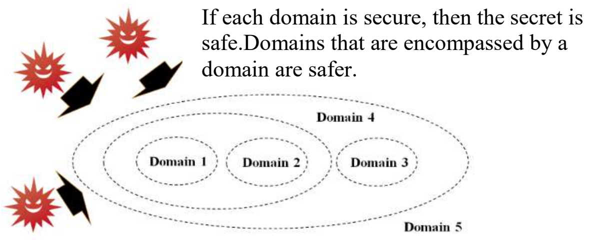
If each domain is secure, then the secret is safe.Domains that are encompassed by a domain are safer.
Domain 4
Domain 1
Domain 2
Domain 3
Domain 5

Fig. 3. Domains and their structure.

Although there are many factors that affect society's ability to resist a virus, it is reasonable to regard them as parameters of the above structure. For example, a country that inhibits entrance to foreigners corresponds to a reduction in the probability of the virus entering the country.

The key idea of ref. [8] is to apply the above perspective to the problem described in Subsection *B*. That is, ref. [8] shows that communications networks, including the Internet, are actually a group of many domains, including DNS domains, DNS zones, and private networks, where security measures are executed by the managers of each domain. This is similar to the framework of countermeasures against a virus explained through Fig. 3, and therefore, a model based on a domain structure is also reasonable for a communications network.

D. Mathematical Formulation

Based on the idea presented in the previous subsection, ref. [8] proposed the mathematical model, $\Lambda = (G, D)$, where G (denoting the communications network) $= (V, E)$ and D denotes the set of domains, i.e., $\{D_1, D_2, \ldots, D_m\}$, where D_i ($i = 1, 2, \ldots, m$) $\subseteq V$.

In particular, a node in G represents a server, router, or other equipment. A link in G represents a logical link between two nodes. D represents the set of DNS domains, DNS zones, private networks, or other sets of independently controlled equipment on the communication network.

The nodes are divided up into three groups. Nodes in the first group are called 'intrusion gates', nodes in the second group are called 'participants', and nodes in the third group are called 'transition nodes'.

Participants represent a set of equipment having shares, and the hacker's target is to reach k participants from one of the intrusion gates through the paths of G. An example of Λ is illustrated in Fig. 4.

Note that in the figures in this paper, filled circles represent intrusion gates, and the other circles show participants or transition nodes. Participants are additionally indicated by arrows like in Fig. 4. If a circle is not filled and not pointed to by an arrow, then it is a transformation node. Domains are illustrated as dotted closed curves.

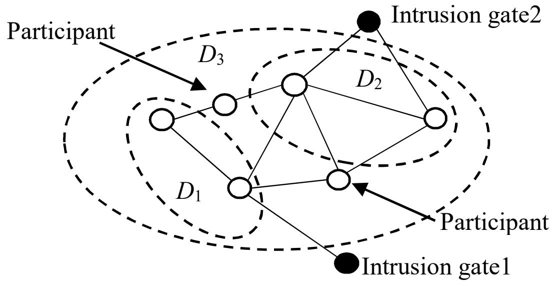

Fig. 4. Example of Λ.

$D_1, D_2, \ldots, D_m$ satisfy the following conditions.

Domain condition 1.
For any $D_i, D_j \in D$, $D_i \subseteq D_j$ or $D_j \subseteq D_i$.
Domain condition 2.
For any D_i, a real number p_i satisfying $0 \leqq p_i \leqq 1$ is assigned.

If a hacker can intrude into domain D_i, we say that 'D_i is open'; else 'D_i is closed'. The events of domains becoming open are assumed to be probabilistically independent. p_i in domain condition 2 is the probability that a hacker can open the corresponding domain.

A hacker must break the security system of every domain on the paths from an intrusion gate to k participants.

E. Measure of Security Strength

Ref. [8] proposed to evaluate the security strength of Λ by the probability of a hacker being able to reach k participants from an intrusion gate. We denote this probability by P_L.

For ease of understanding, we will explain P_L with the help of the example in Fig. 4, where we will assume that $k = n = 2$.

A hacker can intrude into the network shown in Fig. 4 from either of intrusion gate 1 or 2. Let us suppose that the hacker enters through intrusion gate 1. In this case, he succeeds in reaching all participants if domains D_1 and D_3 are open. Next suppose that the hacker enters through intrusion gate 2. In this case, he succeeds in reaching all participants if domains D_2 and D_3 are open. Therefore, P_L is determined by

$$P_L = P(\text{'}D_1 \text{ and } D_3 \text{ are open' or '}D_2 \text{ and } D_3 \text{ are open'})$$

(See Section Ⅱ for $P(\ )$.)

In this case, P_L can be easily evaluated (computed) as follows.

$$\begin{aligned} P_L &= P(\text{'}D_1 \text{ and } D_3 \text{ are open' or '}D_2 \text{ and } D_3 \text{ are open'}) \\ &= P(\text{'}D_1 \text{ and } D_3 \text{ are open'}) + P(\text{'}D_2 \text{ and } D_3 \text{ are open'}) \\ &\quad - P(\text{'}D_1 \text{ and } D_3 \text{ are open' and '}D_2 \text{ and } D_3 \text{ are open'}) \\ &= p_1p_3 + p_2p_3 - p_1p_2p_3 \qquad (1) \end{aligned}$$

If $p_1 = 0.01$ and $p_2 = p_3 = 0.02$, then $P_L = 0.01 \times 0.02 + 0.02 \times 0.02 - 0.01 \times 0.02 \times 0.02 = 0.00596$.

Below, we give a mathematical definition for P_L.

First, we define a random variable X_h for $h = 1, 2, \ldots, m$, satisfying the following condition.

Condition for X_h.
$X_h = 1$ if domain D_h is open and D_h is deleted from Λ.
$X_h = 0$ if domain D_h is closed and D_h and all nodes in D_h are deleted from Λ (because all nodes in D_h do not exist from the viewpoint of the hacker if D_h is closed.)

For example, if $X_1 = 1$, $X_2 = 0$, and $X_3 = 1$, then Fig. 4 is converted into the model in Fig. 5, and the hacker can reach all participants.

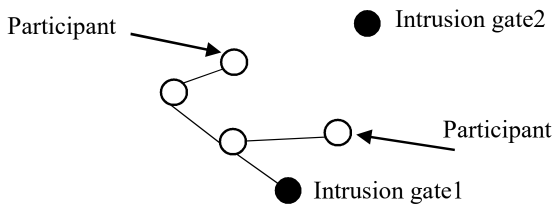

Fig. 5. Converted model.

Second, we define the concept of a connected set.

If $A = \{D_{a_1}, D_{a_2}, \ldots, D_{a_w}\} \subseteq D$ satisfies the following conditions, then A is called a 'connected set'.

Connection condition 1.
If $X_h = 1$ for any $h \in \{a_1, a_2, \ldots, a_w\}$, then k participants and one of the intrusion gates are connected in the corresponding converted model.
Connection condition 2.
If $X_h = 0$ for at least one of $h \in \{a_1, a_2, \ldots, a_w\}$ and any $h \in \{1, 2, \ldots, n\} - \{a_1, a_2, \ldots, a_w\}$, then no participant is connected to any of the intrusion gates. (See Section Ⅱ for subtraction of sets.)

A connection set is a minimal set of domains through which the hacker can reach k participants if these domains are open.

In the case of Fig. 4, {1, 3} and {2, 3} are connection sets.

P_L is defined in terms of the connection sets as follows.

$$P_L = P(\text{all domains in at least one connection set are open})$$

F. Algorithm for Evaluationg P_L

Ref. [8] also proposed an algorithm to evaluate P_L, called the 'truth table algorithm'. This subsection summarizes it.

A 'state' is defined as $(X_1, X_2, \ldots\ X_m)$. The set of all states for a given m is denoted by S_m. For example, S_3 is as follows.

$$S_3 = \{(0, 0, 0), (1, 0, 0), (0, 1, 0), (0, 0, 1), (1, 1, 0), (1, 0, 1), (0, 1, 1), (1, 1, 1)\}$$

If the h-th element of a state is 1, domain D_h is open; else D_h is closed. The structure function $\Phi(X_1, X_2, \ldots, X_m)$ is a function from S_m to {1, 0}.

If k participants and one of the intrusion gates are connected in the corresponding converted model for a given state $(X_1, X_2, \dots, X_m)$, then $\Phi(X_1, X_2, \dots, X_m) = 1$; else $\Phi(X_1, X_2, \dots, X_m) = 0$.

$\Phi(X_1, X_2, X_3)$ for Fig. 5 is the following truth table:

$\Phi(0, 0, 0) = 0$, $\Phi(1, 0, 0) = 0$, $\Phi(0, 1, 0) = 0$, $\Phi(0, 0, 1) = 0$, $\Phi(1, 1, 0) = 0$, $\Phi(1, 0, 1) = 1$, $\Phi(0, 1, 1) = 1$, $\Phi(1, 1, 1) = 1$.

Suppose $Q_h = p_h$ if $X_h = 1$ and $Q_h = 1 - p_h$ if $X_h = 0$.
Now, P_L can be evaluated as follows.

$$P_L = \sum_{s \in S_m'} \left\{ \prod_{h=1}^{n} Q_h \right\} , \qquad (2)$$

where $S_m' \equiv \{s \mid s \in S_m, \Phi(s) = 1\}$.

For Fig. 4, $P_L = p_1(1 - p_2)p_3 + (1 - p_1)p_2p_3 + p_1p_2p_3$. It is easy to see that this is equivalent to the r.h.s. of Eq. (1).

The truth table algorithm for evaluating P_L is listed below.

```
TRUTH TABLE ALGORITHM
INPUT: $\Lambda$
Step 1. Enumerate all states for $\Lambda$.
Step 2. Compute the output of the structure function for
        every state enumerated in Step 1.
Step 3. Compute the r.h.s. of Eq. (2) and output it as the
        evaluation result.
```

G. Problem of the Turth Table Algorithm

The truth table method enumerates all states for fixed m (number of domains), and the number of these states is 2^m. This algorithm is only valid if we have a small number of domains, while problematic if the number of domains becomes large, because the execution time of this algorithm on computer exponentially increases due to the increase of the number of domains. Accordingly, it is an urgent task to develop a faster algorithm for evaluation.

IV. Proposal

A. Key Idea

We define $P_L(\Lambda)$ as the value of P_L for model Λ. If we fix the D_i of Λ to be open, we denote the model by Λ_i^+. If we fix the D_i to be closed, we denote it by Λ_i^-. The following equation is true.

$$P_L(\Lambda) = p_iP_L(\Lambda_i^+) + (1 - p_i)P_L(\Lambda_i^-) \qquad (3)$$

This equation can be obtained by the same logic used to derive the factoring theorem in the reliability engineering field [10].

We call the operation from Λ_i to Λ_i^+ and Λ^- a 'factoring operation'. An example of a factoring operation is shown in Fig. 6. (from here on, the figures will sometimes omit the arrows indicating the participants for simplicity.)

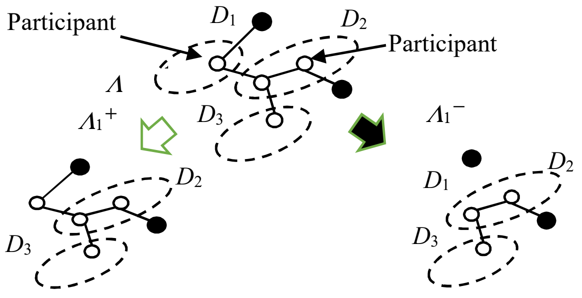

Fig. 6. Example of factoring operation.

Eq. (3) implies that the problem to evaluate P_L for Λ reduces to problems of evaluating $P_L(\Lambda_i^+)$ and $P_L(\Lambda_i^-)$. If these reduced problems are still difficult to solve, then we can apply the factoring operation to Λ_i^+ and Λ^-. The conceptual view of this repeated factoring is illustrated in Fig. 7.

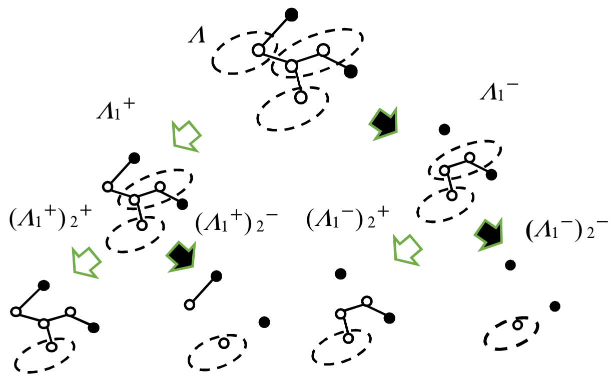

Fig. 7. Repeated factoring operations.

We call each model in this binary tree search a 'stage'.
The factoring operation of this binary search stops if we reach a stage satisfying either of the following stop conditions:

Stop condition 1.
We have no domain with k participants and an intrusion gate connected.

Stop condition 2.
k participants and intrusion gate are not connected even if all domains of the converted model of its stage are assumed to be open.

If a stage, denoted by Λ', satisfies Stop condition 1, then the hacker can aggregate k shares, and $P_L(\Lambda') = 1$ at this stage. On the contrary, if Λ' satisfies Stop condition 2, then the hacker is not able to aggregate k shares, and $P_L(\Lambda') = 0$ at this stage.

Accordingly, if the binary tree search reaches a stage satisfying either of the stop conditions, then no further factoring operation of the model is necessary at this stage.

If we ignore Stop condition 2, then the computation speed for $P_L(\Lambda)$ is obviously proportional to 2^n and its complexity is the same as that of the truth table algorithm.

However, we can cut the size of the tree for the binary tree search after finding stages satisfying Stop condition 2. Therefore, an algorithm based on this idea is expected to be faster than the truth table algorithm.

B. Procedure of Proposed Algorithm

The following recursive procedure for evaluating P_L is based on the idea of the previous subsection.

```
FACT( )
INPUT: $\Lambda$
Step 1. If $\Lambda$ satisfies Stop condition 1, then output 1 and end.
Step 2. If $\Lambda$ satisfies Stop condition 2, then output 0 and
        end.
Step 3. Select a domain $D_i$ randomly.
Step 4. Output $p_i$FACT($\Lambda_i^+$) + (1-$p_i$)FACT($\Lambda_i^-$).
```

C. Motivational Example

Fig. 8 shows a motivational example, where $p_1 = p_2 = p_3 = 0.01$ and $k = n = 2$.

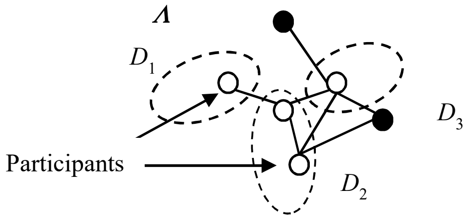

Fig. 8. Example of model.

In Steps 1 and 2 of FACT( ), we can see that Λ satisfies neither stop condition. In Step 3, for example, we select D_2 to be factored and obtain Λ^+ and Λ^- as in Fig. 9.

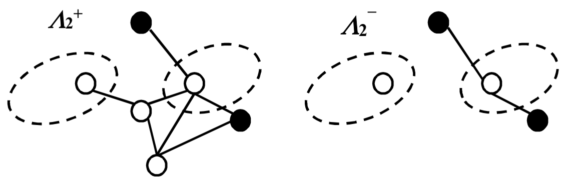

Fig. 9. First factoring operation.

We find that Λ_2^- satisfies Stop condition 2. Therefore, we obtain $P_L(\Lambda_2^-) = 0$. On the other hand, Λ_2^+ can be further factored as in Fig. 10 if we select domain D_1 in Step 3 in the second loop.

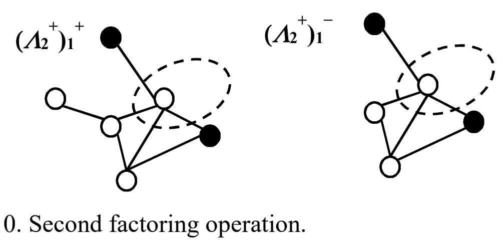

Fig. 10. Second factoring operation.

We find that $(\Lambda_2^+)_1^-$ satisfies Stop condition 2. Therefore, we obtain $P_L(((\Lambda_2^+)_1^-) = 0$, while $(\Lambda_2^+)_1^+$ can be further factored as in Fig. 11.

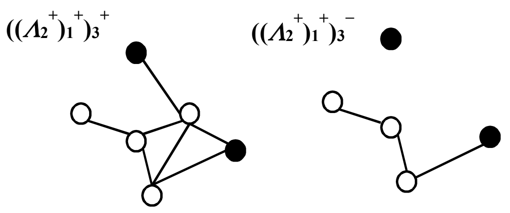

Fig. 11. Third factoring operation.

We find that $((\Lambda_2^+)_1^+)_3^+$ satisfies Stop condition 1, resulting in $P_L(((\Lambda_2^+)_1^+)_3^+) = 1$, and $((\Lambda_2^+)_1^+)_3^-$ also satisfies Stop condition 1, resulting in $P_L(((\Lambda_2^+)_1^+)_3^-) = 1$.Thus, we have

$$
\begin{aligned}
P_L &= P_L(\Lambda) = p_2\mathrm{FACT}(\Lambda_2^+) + (1\text{-}p_2)\mathrm{FACT}(\Lambda_2^-) \\
&= p_2\{p_1\mathrm{FACT}(\Lambda_2^+)_1^+) + (1\text{-}p_1)\mathrm{FACT}(\Lambda_2^+)_1^-)\} + (1\text{-}p_2)\ \mathrm{FACT}(\Lambda_2^-) \\
&= p_2[\ p_1\{p_3\mathrm{FACT}((\Lambda_2^+)_1^+)_3^+ + (1\text{-}p_3)\mathrm{FACT}(((\Lambda_2^+)_1^+)_3^-)\} \\
&\quad + (1\text{-}p_2)\mathrm{FACT}(\Lambda_2^-)]
\end{aligned}
$$

We have already found that $\mathrm{FACT}((\Lambda_2^+)_1^+)_3^+) = \mathrm{FACT}(((\Lambda_2^+)_1^+)_3^-) = 1$, $\mathrm{FACT}(\Lambda_2^-) = 0$, and we know that $p_1 = p_2 = p_3 = 0.01$.Therefore,

$$
\begin{aligned}
P_L &= p_2\ [p_1\{p_3 \times 1 + (1\text{-}p_3) \times 1\}] + (1\text{-}p_2)\ \times 0 \\
&= 0.01 \times [0.01\ \times \{(0.01 \times 1) + (1\text{-}0.01) \times 1\}\} = 0.0001.
\end{aligned}
$$

If we apply the truth table algorithm, we must enumerate $2^3 = 8$ states. However, our proposal requires a relatively compact form like the above.

V. NUMERICAL EXPERIMENTS

A. Environment

We implemented the truth table algorithm and FACT( ) in the following environment:

OS: Windows 10 Home
CPU: Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
2.70 GHz
RAM: 8.00GB Language: C

B. Target Models

The network topology, domains, and participants are illustrated in Fig. 12. The value of p_i is 1.00×10^{-4} for every domain. $k = 2$ and $n = 3$. There are three ways of assigning intrusion gates to Fig. 12. These are illustrated in Fig. 13, 14, and 15.

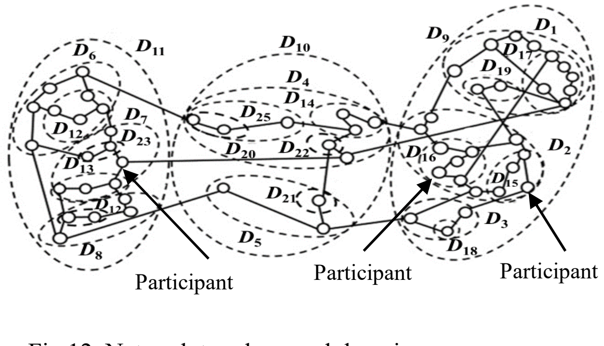

Fig.12. Network topology and domains.

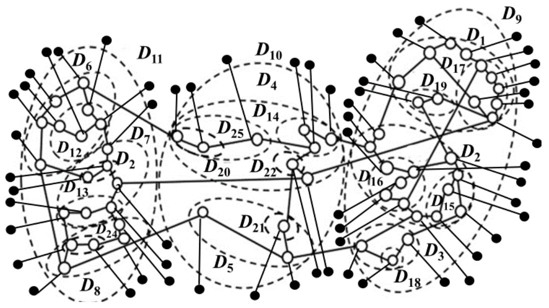

Fig. 13. Pattern 1.

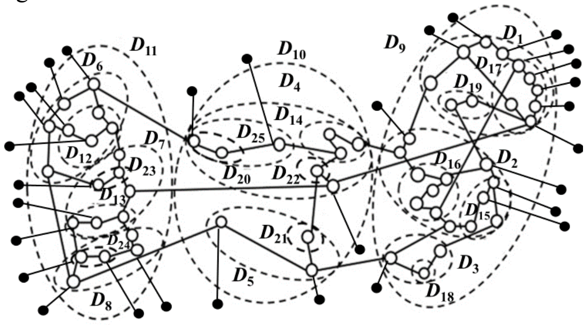

Fig. 14. Pattern 2.

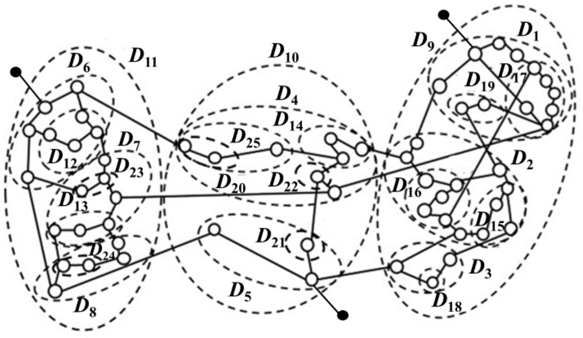

Fig. 15. Pattern 3.

C. *Results and Discussion*

The results of the evaluation are listed below.

$P_L = 1.00\times10^{-16}$ for Patterns 1 and 2. $P_L = 1.00\times10^{-20}$ for Pattern 3. The computation time for Pattern 1 was 118.84 seconds when we used FACT( ) and 4269.73 seconds when we used the truth table algorithm. The computation time for Pattern 2 was 114.88 seconds when we used FACT( ) and 4269.03 seconds when we used the truth table algorithm. The computation time for Pattern 3 was 63.05 seconds when we used FACT( ) and 4251.11 seconds when we used the truth table algorithm.

The results indicate the followings.

1. The proposed algorithm is quite faster than the truth table algorithm.
2. The security strength does not change if we reduce the number of intrusion gates by up to a half.
3. However, the security strength becomes quite stronger if we reduce the number of intrusion gates by more than a half.

Thus, the number of intrusion gates affects the security strength.

If we do not have stop conditions, then the computational complexity of our algorithm is equivalent to existing truth table method. However, if we can avoid one execution of factoring at the stage that ωnumber of domain are still not factored then 2^{ω} number of leaves in our binary tree of FACT( ) are successfully cut. If $\omega = 10$, then 1024 number of leaves can be cut. That's why, we can expect that great reduction of computational complexity is realized by our algorithm than truth table method not only for our examples in this section but also in general cases.

VI. Conclsion

This paper has proposed a new algorithm to evaluate security strength for allocation of shares in the secret sharing scheme. Key idea is to use binary tree search with reasonable stop conditions. These stop conditions are expected to effectively reduce the size of binary tree, and computational complexity is estimated to become greatly smaller than existing algorithm. Numerical examples give evidences to support this estimation.

Future work will include more theoretical analysis of computational complexity of our algorithm, a further improvement of the algorithm, and new applications.

References

[1] A. Shamir, "How to share a secret," Communications of the ACM, vol. 22, no. 11, pp. 612- 613, 1979.
[2] G. R. Blakeley, "Safeguarding cryptographic keys," AFIPS, vol. 48, pp. 313-317, 1979.
[3] N. A, Ebri et al., "Study on secret sharing schemes (SSS) and their applications," International Conference on Internet Technology and Secured Transaction, pp. 40-45, 2011.
[4] P. Dharani et al., "Survey on secret sharing scheme with deduplication in cloud computing," ISCO, 15490813, 2015.
[5] R. H. Shar et al., "A multifactor authentication system using secret splitting in the perspective cloud of things," International Conference on Emerging Trends and Innovation in ICT, INSPEC Accession Number 17029175, 2017.
[6] D. Chen et al., "An efficient verifiable threshold multi-secret sharing scheme with different stages," IEEE Access, vol. 7, pp. 107104 – 107110, 2019.
[7] L. Tan et al., "Weighted secret image sharing for a (*k*, *n*) threshold based on Chinese remainder theorem," IEEE Access, vol. 7, pp. 2169-3536, 2019.
[8] T. Kuwabara et al., "Framework and solution for assigning shares to communication network domains under secret sharing scheme," ITC-CSCC, pp. 330-335, 2020.
[9] A. Hineman et al., "A modified Shamir secret sharing scheme with efficient encoding," IEEE Communications Letters, vol. 26, no. 4, pp. 758-762, 2022.
[10] L. B. Page et al., "A practical implementation of the factoring theorem for network reliability," IEEE Trans. Reliability, vol. 37, no. 3, pp. 259-267, 1988.