Theoretical Computer Science 939 (2023) 237–249

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

# A fault diagnosis method to defend scapegoating attack in network tomography ☆

Xiaojia Xu, Yongcai Wang *, Yu Zhang, Deying Li

*School of Information, Renmin University of China, Beijing 100872, China*

ARTICLE INFO

*Article history:*
Received 17 September 2022
Received in revised form 17 October 2022
Accepted 22 October 2022
Available online 26 October 2022

*Keywords:*
Network tomography
Fault diagnosis
Unobserved cut set
Scapegoating attack
Identifiability

ABSTRACT

The scapegoating attack can cause persistent and inconspicuous performance degradation in network tomography. Defense of scapegoating attack is therefore a critical problem. Theoretically, the ideal defending scheme is to add monitoring paths to make all the links in the network be identifiable. This requires very high monitoring cost, which is unaffordable. To overcome this problem, this paper proposes a diagnosis-based defending scheme for scapegoating attack, which diagnoses scapegoating attack when problematic links are detected by network tomography. The latent fact is that a scapegoating attack can be launched only when the link set manipulated by the attacker cuts the probing paths going through the scapegoat links and is not traversed by any monitoring path. This cut set is called unobserved cut set (UCS). To defense, we propose to find the UCS and add the minimum number of probing paths to traverse the UCS, so that the condition of scapegoating attack is broken and the attacking links can be detected if any scapegoating attack exists. A minimum set cover model is proposed to select the least number of defense links to cover the UCS, and a polynomial time algorithm is proposed to generate the least number of probing paths to go through the selected defense links. Evaluations on various network dataset show the effectiveness of the proposed attack and defense strategies.



## 1. Introduction

With the continuous development of computer network technology, knowing the internal states of the network (e.g., bandwidth, packet loss rate, link delay) timely and accurately is an important requirement in network management. Instead of directly measuring the elements within the network, *network tomography* which uses end-to-end path measurements to infer the internal state of the network [2][3], becomes a promising solution [4][5]. Network tomography deploys a set of monitors in the network [6], and measures only the end-to-end path performances between the monitors, and then infers the internal states of links and nodes by solving state recovering functions [7][8][9][10]. It avoids the issues such as high internal measurement overhead, high measurement cost of the direct measurement methods. A critical problem in network tomography is the "identifiability" problem, which indicates whether the internal states of the network can be uniquely recovered by the end-to-end external path measurements.

☆ This is an enhanced and extended version of a paper [1] presented in The 16th International Conference on Algorithmic Aspects in Information and Management (AAIM 2022). This work is partially supported by the National Natural Science Foundation of China Grant No. 12071478, 61972404. Public Computing Cloud, Renmin University of China.

* Corresponding author.
*E-mail addresses:* xuxiaojia@ruc.edu.cn (X. Xu), ycw@ruc.edu.cn (Y. Wang), 2020104230@ruc.edu.cn (Y. Zhang), deyingli@ruc.edu.cn (D. Li).

https://doi.org/10.1016/j.tcs.2022.10.029
0304-3975/

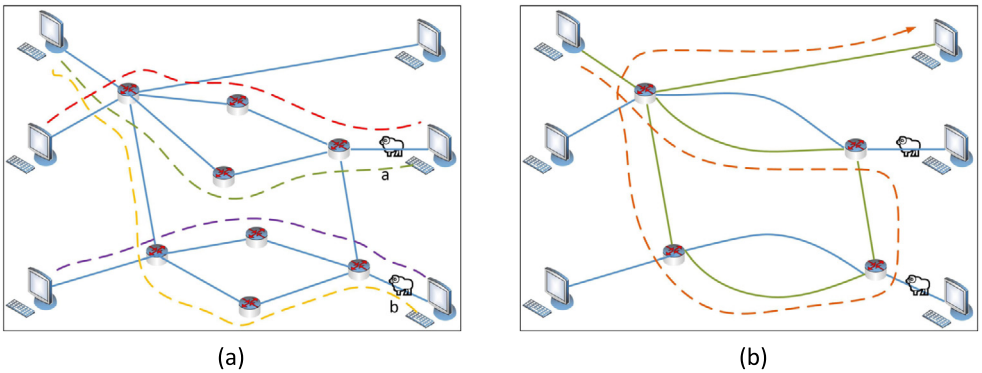



**Fig. 1.** An example of the defense method.

Recently, the risk of being attacked is noticed when the identifiable property is not satisfied. *Scapegoating attack (SA)* [11] refers to a kind of attack, that when an attacker manipulates a set of links to inject attacks (such as inject delay or discard packets), not only the network performances will be degraded, but also the network tomography will be misled to guilt a set of normal links as scapegoats of the attack. Chiu et al. [12] consider to degrade the performance of the network by injecting delays to some path measurements and network tomography cannot localize the attackers. They introduce chosen-victim, maximum-damage and obfuscation scapegoating attacks. Recent work [13] proposes the conditions to successfully launch scapegoating attack. The presence of backdoor infected routers [14] and node-capture attacks [15] can be utilized to carry out above scapegoating attacks by affecting the packet delivery and the path measurement.

The scapegoating attack can cause persistent and inconspicuous performance degradation. The ideal defense scheme is to insure the probing paths of network tomography satisfying the "identifiability condition" [9], which requires *the rank of the routing matrix is equal to the number of links* [7]. However, the number of links can be very large, the identifiability condition is hard to be satisfied, unless the number of probing paths is not less than the number of links, which need very high measurement costs. Efficient methods to defend against scapegoating attack without greatly increasing measurement cost are highly desired. To address this problem, this paper proposes a highly efficient fault diagnosis based scheme for defending against scapegoating attacks. It doesn't require the probing paths to satisfy the identifiability condition. Instead, when network tomography detects problematic links, we propose to use very low probing cost to examine whether the problematic links are scapegoats or are real network problems.

Fig. 1 shows an intuitive example of our defense method. Fig. 1(a) shows a sample network with two problematic links $a$ and $b$ detected by network tomography. At this time, we need fault diagnosis method to judge whether these two links are actually problematic or they are scapegoats. Fig. 1(b) gives the path generated by our algorithm to verify whether the link is really problematic or under scapegoating attack. The key behind the scheme is that we investigate the necessary and sufficient condition to defense the scapegoating attack. When problematic links are discovered by network tomography, we propose to add the minimum number of probing paths to break the conditions of scapegoating attack on the problematic links, which is formulated as a minimum cost defending problem. The main contributions of this paper are as follows.

- Firstly, the necessary and sufficient condition to launch scapegoating attack is proposed. It is that the attacker manipulates all links in a minimal cut set of the probing paths that pass through the scapegoat links but are not observed by any probing path. Such link sets are called unobserved cut set (UCS).
- This paper proposes to find the UCS and add the minimum number probing paths to traverse them, so that the condition of scapegoating attack is broken and the attacking links can be detected if any scapegoating attack exists.
- A minimum set cover model is proposed for selecting the least number of defense links to cover all the UCS; a greedy approximation algorithm with $H_K$ ratio is proposed to solve the defense link selection problem; and a polynomial time edge-disjoint path generation algorithm is proposed to generate defending paths to traverse these links-to-defend.
- Extensive verification on real network datasets shows the effectiveness of the proposed defense strategies.

## 2. Related work

### 2.1. Network tomography

Network tomography has been used for link measurement and fault location in wired [4] and wireless networks [5]. The early work focused on finding the most likely network state from the given measured path values [8][9], which is called additive network tomography. [16] studies link delay distribution estimation from unicast edge measurements. [17] studies to infer the internal topological structure based on end-to-end multicast and unicast observations; [18] introduces the detection mode of dual broadcasting. [8] proposes the efficient algorithm MMP to place a minimum number of monitors to identify all link metrics. [9] studies the algorithm STPC for constructing linearly independent measurement paths.

The supervision of some network systems does not always need to accurately grasp the specific values such as delay packet loss rate, sometimes only need to grasp the status of services or devices such as congestion location and fault location [19][20], their measurement values are binary. In this case, modeling the problem with a system of Boolean equations becomes more convenient because each unknown parameter will have only two states [21][22]. Therefore, Boolean network tomography aims to infer the binary states of unknown parameters from end-to-end Boolean metrics [23][24]. Boolean network tomography is a widely studied branch of network tomography. In this paper, we focus on additive network tomography [9].

### 2.2. Guaranteeing identifiable is difficult

Identifying all link metrics uniquely requires a routing matrix having rank equal to the number of links, which is however difficult to satisfy for the number of links is large, and the dependence of topology problem [8]. Rank deficiency of routing matrix will cause solution ambiguity. A branch is to investigate the placement of monitors and the design of the probing paths to ensure the identification property.

Ref. [25] proves that if the network is directed (links have different metrics in different directions), link metrics cannot be all identifiable unless every non-isolated node is a monitor. When the network is undirected, [26] derives the first necessary and sufficient conditions for identifying all link metrics when probing paths may contain cycles. But in network tomography, paths containing circles are generally avoided. Further, [27] characterizes the minimum number of measurement paths needed to identify additive and non-additive link metrics. They allow the probing paths to contain repeated links.

Under the constraint that the probing paths must be cycle-free, [8] proves the identifiable conditions of link metrics. The conditions show that the identifiability property depends on the network topology. In sparse networks, guaranteeing identifiability needs to select a large number of monitors, which is generally not practical.

Seeing the difficulty, [6] considers partial identification problem using a limited number of monitors, which maximizes the number of identifiable links by optimizing the placement of a limited number of monitors. [23] considers the optimal monitor placement for localizing node failures. [28] considers the optimization problem of monitor placement when the network may change topologies. On the other hand, given the set of monitors, the optimization of the measurement paths is studied in [9,29].

### 2.3. Scapegoating problems and existing defense strategies

Methods like Pseudo-inverse, WCF estimator [3] and congested link identification algorithm [30] are generally used to recover the link states when identification is not guaranteed. When the identifiable property is not satisfied, the risk of scapegoating attack is noticed. [11] introduces the scapegoating problems. They introduce chosen-victim, maximum-damage and obfuscation scapegoating attacks. [12] considers to degrade the performance of the network by injecting delays to some path measurements and network tomography cannot localize the attackers.

Existing defense strategies are usually deployed on the host system to directly detect anomalies of specific victims. For example, packet marking and filtering mechanisms can mark legitimate packets so that the victimized edge router can filter the attack stream [31]. There is an IP backdating mechanism that can trace back to the real source of the forged IP packet [32]. Traffic monitoring is used to detect the forwarding of abnormal packets [33], and new strategies are designed to detect routers that discard or misroute packets [34]. Currently, the study of conditions and strategies to defense scapegoating attack remains open.

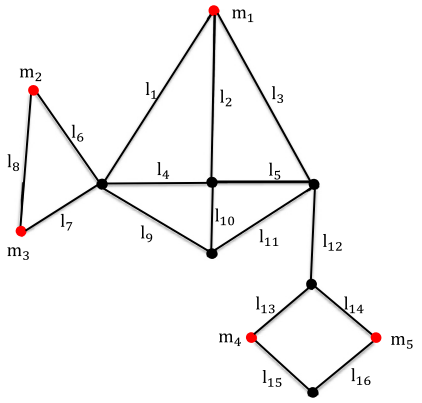
## 3. Problem formulation

### 3.1. Network tomography model

We consider a network modeled as a weighted, undirected graph $G = (V, L, X)$. $V$ and $L$ are the sets of nodes and links, $|V| = n$ and $|L| = m$. Set $X$ represents the link weights, where $x_i \in X$ is an unknown link metric that describes the link $i$'s performance, such as latency and loss rate. We assume the measures of these link metrics are additive. This is a canonical model for representing important performance measures [8,9,23]. In the network, a subset of vertices used for injecting and extracting probing packets is defined as the set of monitors $M = \{m_i\}$. Monitor placement algorithms and probing path generation algorithms in network tomography can be referred to [9][8].

A probing path $p_i$ for network tomography is defined as a sequence of links that starts from a source monitor $s_i$ and ends at a destination monitor $d_i$. The end-to-end measurement of this probing path is denoted by $y_i$. $P = \{p_i\}$ is the set of probing paths and $Y = \{y_i\}$ denotes the end-to-end measurements of these paths. Routing matrix $R = \{r_{i,j}\}_{p_i \in P, l_j \in L}$ models how the probing paths traverse the links in $L$. $r_{i,j} = 1$ if a link $l_j \in L$ is on the path $p_i \in P$ and $r_{i,j} = 0$ otherwise. Network tomography is to solve the equation $R\hat{x} = Y$ to find a solution $\hat{x}$ that represents the estimated link metrics [8].

Note that this linear equation has a unique solution when the routing matrix $R$ has full column rank, i.e., $\text{Rank}(R) = m$, which is the unique identification condition [7]. However, since $m \gg n$, it is generally difficult to generate enough probing paths to satisfy the identification condition. For such difficulty, network managers generally adopt the routing matrix with $\text{Rank}(R) < m$. In such case, Pseudo-inverse is generally used [11][12] to estimate $\hat{x}$ by:




$$
\begin{aligned}
p_1 &= \{l_1, l_6\} \\
p_2 &= \{l_2, l_4, l_7\} \\
p_3 &= \{l_1, l_9, l_{11}, l_{12}, l_{13}\} \\
p_4 &= \{l_3, l_{12}, l_{14}\} \\
p_5 &= \{l_8\} \\
p_6 &= \{l_6, l_9, l_{11}, l_{12}, l_{13}\} \\
p_7 &= \{l_6, l_4, l_5, l_{12}, l_{14}\} \\
p_8 &= \{l_7, l_4, l_5, l_{12}, l_{13}\} \\
p_9 &= \{l_7, l_9, l_{10}, l_5, l_{12}, l_{14}\} \\
p_{10} &= \{l_{15}, l_{16}\}
\end{aligned}
$$

**Fig. 2.** A network with five monitors selected by MMP [35] algorithm and ten probing paths. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

$$\hat{x} = (R^T R)^{-1} R^T Y \tag{1}$$

Without loss of generality, we consider the link metric is additive, such as link delay. Then network tomography determines states of links by following method:

**Definition 1** *(Estimated link states).* Given estimated link delay $\hat{x}$, a link may be divided into three states:

$$\psi(l_i) = \begin{cases} normal & \text{if } \hat{x}_i < \beta_{min} \\ uncertain & \text{if } \beta_{min} \leqslant \hat{x}_i \leqslant \beta_{max} \\ problematic & \text{if } \hat{x}_i > \beta_{max} \end{cases} \tag{2}$$

where $\beta_{min}$ is a threshold to find normal links. $\beta_{max}$ is a threshold to detect problematic, i.e., problematic links. When problematic links are detected, network tomography generally requires more diagnosis method to check and to fix the detailed problem.

### 3.2. Scapegoating attack model

Consider the attacker hacks a set of links, say $L_m \subset L$ is manipulated by the attackers. Attacker injects delays onto these links to affect the probing paths that pass through these links. The attackers also hope to hide their attacks by let network tomography to wrongly detect some other links as problematic. The wrongly detected problematic links are called "scapegoats". The measurement model under attack is:

$$R(x + \Delta x) = Y' \tag{3}$$

where $\Delta x_i > 0$ for $l_i \in L_m$ and $\Delta x_i = 0$ for $l_i \notin L_m$.

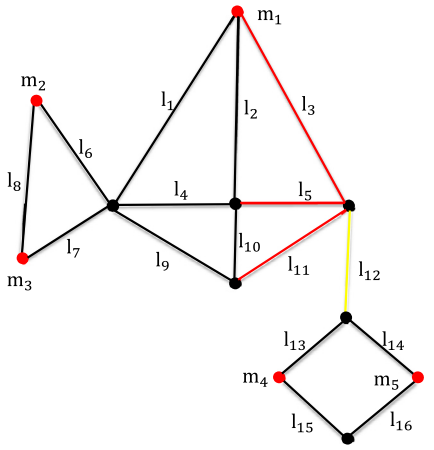
Refs. [11][12][13] show that scapegoating attack can be successfully launched when the probing paths don't satisfy identifiability condition. Let $L_s \subset L$ be the set of scapegoats. There should be $L_s \cap L_m = \emptyset$, since attacking links should not be discovered. The **scapegoating attack** is called successful launched if:

$$\begin{cases} \hat{x}_i \leq \beta_{min} \text{ for } l_i \in L_m \\ \hat{x}_i > \beta_{max} \text{ for some } l_i \in L_s \end{cases} \tag{4}$$

where $\hat{x} = \{\hat{x}_1, \hat{x}_2, \cdots \hat{x}_n\}$ is the solution of $\hat{x} = (R^T R)^{-1} R^T Y'$, which is the network tomography results under attack.

In Fig. 2, we using a simple network with five monitors to illustrate the scapegoating attack. The red markers are the monitors which are selected by MMP algorithm [8]. In order to identify the link metrics, ten probing paths are constructed among monitors, as listed in the right part of Fig. 2.

Fig. 3 shows the result of scapegoating attack. The manipulated links are $l_3, l_5$ and $l_{11}$ and the scapegoating link is $l_{12}$. The attack misleads the network tomography to conclude that link $l_{12}$ is a problematic link and the administrator cannot identify the attacking links by simply checking the "problematic link" $l_{12}$. This not only greatly degrades the network performance, but also imposes high difficulty to detect the true faults.




scapegoating link : $l_{12}$

manipulated link : $l_3\ l_5\ l_{11}$

true metric :
$x_1 = 10\ x_2 = 10\ x_3 = 1480\ x_4 = 10$
$x_5 = 1450\ x_6 = 10\ x_7 = 10\ x_8 = 10$
$x_9 = 1360\ x_{10} = 10\ x_{11} = 1460\ x_{12} = 10$
$x_{13} = 10\ x_{14} = 10\ x_{15} = 10\ x_{16} = 10$

estimated metric :
$\widehat{x_1} = 10\ \widehat{x_2} = 30\ \widehat{x_3} = 20\ \widehat{x_4} = 0$
$\widehat{x_5} = 0\ \widehat{x_6} = 10\ \widehat{x_7} = 0\ \widehat{x_8} = 10$
$\widehat{x_9} = 0\ \widehat{x_{10}} = 20\ \widehat{x_{11}} = 0\ \widehat{x_{12}} = 1480$
$\widehat{x_{13}} = 10\ \widehat{x_{14}} = 0\ \widehat{x_{15}} = 20\ \widehat{x_{16}} = 0$

**Fig. 3.** The result of the scapegoating attack.

## 4. Defense strategy

It is necessary to design defense strategy to avoid scapegoating attack. We firstly investigate the conditions for successful defense, and then propose an efficient, minimum cost fault diagnosis method to examine whether a detected "problematic link" is a scapegoat, so as to discover the attacking links.

### *4.1. What to defend?*

Note that the probing paths can be classified into three categories.

**Definition 2** *(Three types of paths).* The probing paths in $P$ are classified as.

- $P_m \subseteq P$ is the **manipulated path set**. Each path in it passes through at least one manipulated links.
- $P_n \subseteq P$ denotes the **normal path set**. All paths in it doesn't pass through any manipulated link.
- $P_s \subseteq P$ is the **scapegoating path set**, which contains measurement paths that pass through at least one scapegoat link.

The original delay of an attacked link is $x_i$ and the injected delay is $\Delta x_i$. The increased delay for a path $p_i \in P_m$ is further denoted by $\Delta t_i$, where $\Delta t_i = \sum_{l_j \in p_i} \Delta x_j$. Since the injected delays only affect the manipulated path set, we can easily get the following Lemma 1.

**Lemma 1.** *Given $P, L_s, L_m$, if delays are injected onto links in $L_m$, there must be $\Delta t_i \geqslant 0$ for $p_i \in P_m$ and $\Delta t_i = 0$ for $p_i \in P_n$.*

**Proof.** For each $p_i \in P$, if $\Delta t_i \geqslant 0$, there is at least one link $l_j \in p_i$, $l_j \in L_m$. The path $p_i$ passes through at least one manipulated links, therefore, $p_i \in P_m$. For each $p_i \in P$, if $\Delta t_i = 0$, there is no link in $p_i$ belong to $L_m$. The path $p_i$ doesn't pass through any manipulated link, therefore, $p_i \in P_n$. □

Then we separate the roles of link sets according to their properties on the probing paths.

**Definition 3** *(Cut set of paths).* A cut set $C$ of a path set $P$ is a set of links in $P$ that for every path $p_i \in P$, the path passes through at least one link in $C$. In other words, when we cut all the links in $C$, all the paths in $P$ will be cut.

**Definition 4** *(Unobserved cut set (UCS) of $l_s$).* An unobserved cut set of $l_s$ is a set of links that cut the paths that pass through $l_s$ and there is no other probing path that passes through any link in this set.

**Problem 1** *(Diagnosis-based scapegoating defense).* The following problem is specially considered. When network tomography detects a link set $L_s$ is problematic, how to use the minimum additional probing costs to check out whether their states are truly problematic, or they are scapegoats of some attackers.

Suppose a set of scapegoat links $L_s$ are reported problematic, i.e., $\hat{x}_i > \beta_{max}$, for $l_i \in L_s$, where $\hat{x} = \left(R^T R\right)^{-1} R^T Y$ is the detected link states by current routing matrix $R$, constructed using probing paths $P$. But the true states of these links are

normal. Our goal is to add the minimal number of defending paths $P_d$ to construct a new routing matrix $R' = [R, R_d]^T$, where $R_d$ is the routing matrix of $P_d$ to recover the true states of links in $L_s$. By adding $P_d$, the recovered link states are $\hat{x'} = \left(R'^T R'\right)^{-1} R'^T Y'$. $P_d$ is called an effective defense, if the true states of the scapegoat links are recovered. The minimal path defending problem can be stated as follows:

**Problem 2** *(Minimal path defending problem).* The minimal path defending problem for scapegoating attack is to find the minimal number of additional probing paths such that:

$$\min |P_d| \quad s.t. \begin{cases} R' = [R, R_d]^T, Y' = [Y, Y_d]^T \\ \hat{x}' = \left(R'^T R'\right)^{-1} R'^T Y' \\ \hat{x}'_i \leq \beta_{\max}, \forall l_i \in L_s \end{cases} \tag{5}$$

Note that in (5), the original routing matrix $R$ is copied, so the path constraints in original $P_n$ and $P_m$ are still satisfied. The key problem is how to design the defending paths.

### 4.2. Key observations

For defense, although the exact locations of $L_m$ are not known, we know that they must cover an unobserved cut set of $P_s$ [12][13]. For a link $l_i$ in $L_s$, there maybe multiple unobserved cut sets that can cut the probing paths passing through $l_i$. We denote the unobserved cut sets for $l_i \in L_s$ as $\{C_{i,1}, C_{i,2}, \cdots, C_{i,n_i}\}$.

**Lemma 2** *(Breaking an unobserved cut set for $l_i \in L_s$). If we add a probing path $p$ to go through any link in a cut set $C_{i,j}$ but not the link $l_i$, then the injected delays on the cut set $C_{i,j}$ can no longer be attributed to $l_i$.*

**Proof.** Only when all the links in an unobserved cut set of $l_i$ are manipulated by the attacker, can the scapegoating attack to $l_i$ be successfully launched. Therefore, if a link in the unobserved cut set $C_{i,j}$, denoted by $l_k$ has been monitored by an added $p$, the added delay to this link will increase the delay of the new path $p$. Since $p$ doesn't go through $l_i$, the increased delay cannot be attributed to $l_i$. We say the unobserved cut set $C_{i,j}$ is broken by the added path $p$. □

**Theorem 1** *(Necessary & sufficient condition to defend $L_s$). Given $G$, $P$, and $L_s$, to discover whether each link $l_i \in L_s$ is truly problematic or not, we need to add new probing paths $P_d$ which don't go through $l_i \in L_s$, but make each unobserved cut set of $l_i$ have at least one link be passed by at least one path in $P_d$.*

**Proof.** Necessity: From Lemma 2, a unobserved cut set of $l_i$ is broken if one of its link is passed through by an added path. Since every cut set may initial an attack to $l_i$, we need to break all cut sets of $l_i$ to protect $l_i$. So $l_i$ is protected only if all its unobserved cut sets are broken.

Sufficiency: When all the unobserved cut sets are broken, the injected delays on any cut set cannot be attributed to $l_i$. So the true state of $l_i$ will be recovered after adding the defending paths. □

### 4.3. Defense methodologies

For each link $l_i \in L_s$, there maybe many unobserved cut sets that can cut the paths going through $l_i$. Let $P_{l_i} \in P_s$ denote the set of paths that go through $l_i$. Let $K$ denote the number of paths in $P_{l_i}$. Since any unobserved cut set of $P_{l_i}$ must cover an *unobserved minimal cut set (UMCS)* of $P_{l_i}$, so in order to break all unobserved cut sets for $P_{l_i}$, we must break all the UMCS of $P_{l_i}$.

There maybe a large amount of link combinations can generate UMCS of $P_{l_i}$. An example is shown in Fig. 4, in which $l_i = (15, 18)$. $P_{l_i} = \{P_1, P_2, P_3\}$. Let $\mathbb{C}$ be the set of UMCS. Then selecting one link on each path will form an unobserved minimal link cut, so $|\mathbb{C}| = 6 \times 6 \times 7 = 252$. From Fig. 4(a), we can also see that by selecting $P_4$ as a defending path, it will break all the UMCS in $\mathbb{C}$. So adding $P_4$ as a defending path will reveal $l_i$'s true state.

To overcome the large number of UMCS, we propose a probing path segmentation method to merge the links and to reduce the size of $\mathbb{C}$. Suppose the $K$ paths of $P_{l_i}$ are intersecting at some vertices. The intersecting vertices cut each path into several path segments. The segments of the $j$th path are named by $S_j = \{s_{j,1}, s_{j,2}, \cdots, s_{j,n_j}\}$, where $n_j$ is the number of segments on path $j$. Each segment contains several sequentially connected links on a common path. If a defending path passes through a segment, it passes through all links in the segment.

Algorithm 1 gives the algorithm of path segmentation. The matrix of addition and subtraction between two paths reveals whether an edge is passed by both paths simultaneously. For path $p_i$, $p_j$ and link $l_k$, the addition matrix $add_{i,j,k} = 2$ if link $l_k$ is on both path $p_i$ and path $p_j$; $add_{i,j,k} = 1$ if link $l_k$ is on either path $p_i$ or path $p_j$; $add_{i,j,k} = 0$ if link $l_k$ is on neither

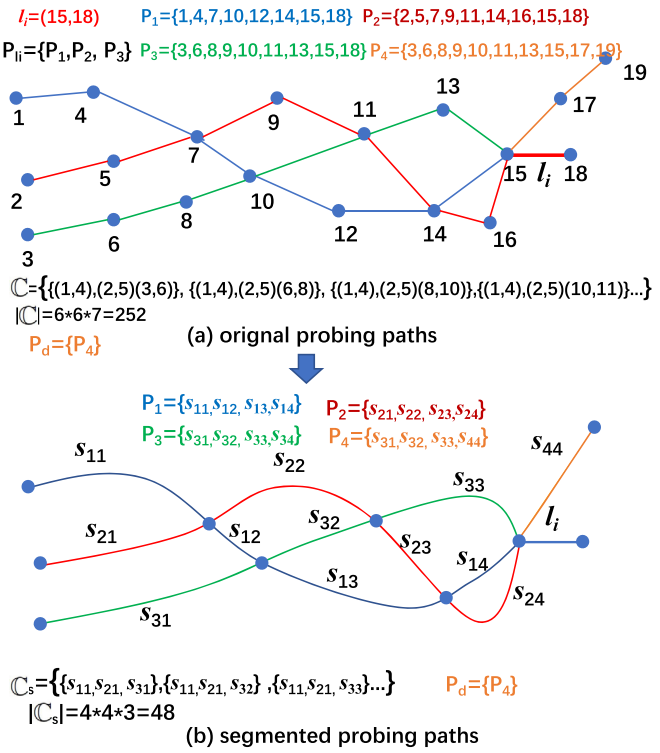



**Fig. 4.** Segments of $P_{l_i}$ and the design of defending path.

**Algorithm 1:** Path segmentation algorithm: $S = PathSeg(R, P)$.

**Input:** $R$, $P$
**Output:** $S$
Initialize $S = \emptyset$ ;
**for** $(i = 1; i \le |P| - 1; i++)$ **do**
  **for** $(j = i + 1; j \le |P|; j++)$ **do**
    **for** $(k = 1; k \le size(R, 2); k++)$ **do**
      $add\{i, j\}(1, k) = R(i, k) + R(j, k)$;
      $sub\{i, j\}(1, k) = R(i, k) - R(j, k)$;
**for** $(i = 1; i \le |P| - 1; i++)$ **do**
  **for** $(j = i + 1; j \le |P|; j++)$ **do**
    **for** $(k = 1; k \le |p_i| - 1; k++)$ **do**
      **if** $add\{i, j\}(1, p_i(k)) \ne add\{i, j\}(1, p_i(k+1))$ **then**
        $PathSlice\{i\}(k) = 1$;
      **if** $sub\{i, j\}(1, p_i(k)) \ne sub\{i, j\}(1, p_i(k+1))$ **then**
        $PathSlice\{i\}(k) = 1$;
**for** *each path* $p_i \in P$ **do**
  count the number of path slice $ps_i$ in $p_i$;
  **for** $(j = 1; j \le ps_i + 1; j++)$ **do**
    derive the segment $s_{ij}$ of path $p_i$ from $R$ and $P$;
return $S$ ;

path $p_i$ nor path $p_j$. Similarly, the subtraction matrix $sub_{i,j,k} = 1$ if link $l_k$ is only on path $p_i$; $sub_{i,j,k} = 0$ if link $l_k$ is on both path $p_i$ and path $p_j$, or on neither path $p_i$ nor path $p_j$; $sub_{i,j,k} = -1$ if link $l_k$ is only on path $p_j$. For link $l_a$ and $l_b$, if link $l_a$ and $l_b$ are on the same path set, the addition matrix and the subtraction matrix will be the same, $add_a = add_b$ and $sub_a = sub_b$. If two adjacent edges on a path are not passed by the same paths, a slice is added between them. We count the number of slices on each path and the number of segments on each path is the number of slices plus one. Finally, we derive segment set $S$ by using slice matrix and path matrix.

Fig. 4(b) shows an example of path segmentation. The three paths passing through $l_s$ are divided into path segments by the intersecting vertices, i.e., $S_1 = \{s_{11}, s_{12}, s_{13}, s_{14}\}$, $S_2 = \{s_{21}, s_{22}, s_{23}, s_{24}\}$, $S_3 = \{s_{31}, s_{32}, s_{33}\}$. The cut set formed by path segments is denoted by $\mathbb{C}_s$.

**Definition 5** *(Segment cut).* By selecting one path segment from each path in the path set $P_{l_i}$, a segment cut to the path set $P_{l_i}$ is formed.

The number of path segments on a path is much less than the number of links. The number of segment cuts of $P_{l_i}$ is at most $n_1 \times n_2 \times \cdots \times n_K$, which is much less than the number of UMCS, i.e., $|\mathbb{C}_s| \ll |\mathbb{C}|$.

**Lemma 3.** *Every minimal link cut set of $P_{l_i}$ must be covered by a segment cut.*

**Proof.** Since every unobserved minimal link cut set must contain links cutting every path of $P_{l_i}$, each link must also be in a path segment. The path segments where these links are located form a segment cut to $P_{l_i}$. So every minimal link cut set has a mapped segment cut. Therefore, every minimal link cut set must be covered by a segment cut. □

**Lemma 4** *(Defense policy). To break all the UMCS for $P_{l_i}$, i.e., to have any UMCS traversed by a defending path, we only need to insure every segment cut having a path segment to be traversed by a defending path.*

**Proof.** From Lemma 3, every unobserved minimal link cut set of $P_{l_i}$ is covered by a segment cut. So if we insure every segment cut having a path segment to be traversed by a defending path, then every minimal link cut set has at least one link be traversed by a defending path. The necessary condition for $l_i$ being a scapegoat is violated. So that by adding the defending path, $l_i$ cannot be a scapegoat and its true state will be recovered. □

### *4.4. Minimum cost defense for single scapegoat link*

We will show that when there is only one scapegoat link, only one defending path is needed.

**Proposition 1** *(The optimal defense policy for one scapegoating link). To recover the true state of a link $l_i \in L_s$, we only need to add one probing path $P_d$ to go through all path segments of **one specific path** in $P_{l_i}$ except the link $l_i$. Then the true state of $l_i$ can be recovered by (5).*

**Proof.** When a defending path passes through the segments of a specific path in $P_{l_i}$, all the minimal link cut sets of $P_{l_i}$ are broken. So the manipulated links can no longer attribute the injected delays to $l_i$ and $l_i$'s true state will be recovered by the new routing matrix. □

Fig. 4 shows the example. To recover the true state of $l_i$, only one defending path $P_d = P_4$ is needed, which goes through $s_{31}, s_{32}, s_{33}$ (for this path has the least number of segments), but not to go through $l_i$.

### *4.5. Minimum cost defense for multiple links*

Suppose there are $L$ links in $L_s$. An intuitive way is to consider each problematic link individually, which needs to add $L$ defending paths. We further show that the problematic links can also be considered jointly.

Consider there are totally $K$ paths in the path set $P_s$. Similar to Fig. 4, each path is divided into several path segments by the common vertices of these paths. Let's denote the path segments as $S_1 = \{s_{1,1}, \cdots, s_{1,n_1}\}, \cdots, S_K = \{s_{K,1}, \cdots, s_{K,n_K}\}$. In order to defend all the links in $L_s$, all the segment cuts of $P_s$ need to be broken. By selecting one path segment from each path, a segment cut to the path set $P_s$ can be formed. Using this method, all minimal segment cuts for $P_s$ can be found, which is denoted by $\mathbb{C}_s$.

Given the segment cut set $\mathbb{C}_s$, to design the minimum number of defending paths, we propose to find a segment set $\mathbb{S}$ with the minimum cardinality, such that every segment cut $C_i \in \mathbb{C}_s$ contains at least one segment in $\mathbb{S}$. Then $\mathbb{S}$ is called a minimum cover of $\mathbb{C}_s$.

This problem is a typical set cover problem, which is NP-hard. A greedy algorithm is proposed to address this minimum cut set cover (MCSC) problem. Let $S$ be the set of all path segments in $\mathbb{C}_s$. Let $U$ be all uncovered segment cuts in $\mathbb{C}_s$.

In the algorithm, if a path segment $s \in S$ appears in $n_s$ segment cuts in $U$, $n_s$ reveals the covering utility of the path segment $s$. In each iteration, the segment with the largest covering utility is selected and is put in $\mathbb{S}$, until all segment cuts in $\mathbb{C}_s$ have been covered. gMCSC has an $H_K$ approximation ratio [36], where $K$ is the largest number of cut sets that share one common path segment in $S$.

**Algorithm 2:** Greedy minimum set cover: $\mathbb{S}$=gMCSC($\mathbb{C}_s$).

**Input:** $\mathbb{C}_s$
**Output:** $\mathbb{S}$
Initialize $U = \mathbb{C}_s$, $\mathbb{S} = \emptyset$, $S$= all path segments in $\mathbb{C}_s$ ;
**while** *($U$ is not empty)* **do**
  select $s$ in $S$ that covers the most number of sets in $U$ ;
  add $s$ to $\mathbb{S}$ ;
  remove the segment cuts covered by $s$ from $\mathbb{U}$ remove $s$ from $S$ ;
return $\mathbb{S}$ ;

**Lemma 5** *(Approximation ratio of gMCSC). Let $K_s$ be the number of segment cuts in $\mathbb{C}_s$ that have common path segment $s$. Let $K = \max_{s \in S} K_s$ be the largest number of segment cuts that share a common path segment. Let $H_K = \sum_{i=1}^{K} 1/i \approx \ln K$, then the gMCSC algorithm returns $\mathbb{S}$ which has at most $H_K$ times segments than the optimal number of segments to make each segment cut in $\mathbb{C}_s$ has at least one segment in $\mathbb{S}$.*

### 4.6. Minimum number defending path generation

From Lemma 4, since $\mathbb{S}$ covers all segment cuts $\mathbb{C}_s$, in order to break all the segment cuts in $\mathbb{C}_s$, defending paths only need to be added to go through all the path segments in $\mathbb{S}$ and don't go through any link in $L_s$. We want to add the minimum number of defending paths to achieve this goal.

In network tomography, a probing path is not required to be a simple path, which can traverse the same edge more than one time. So a polynomial time algorithm is proposed to generate the minimum number of defending paths to go through all the path segments in $\mathbb{S}$ but no links in $L_s$.

**Algorithm 3:** Defending path generation algorithm: $P_d = PathGen(G, M, L_s, \mathbb{S})$.

**Input:** $G, M, L_s, \mathbb{S}$
**Output:** $P_d$
1 $G = G \setminus L_s$ and suppose it has $F$ components ;
2 **for** $(l = 1; l \le F; l++)$ **do**
3  select a pair of monitors $m_j, m_k$ $\mathbb{S} \neq \emptyset$ and suppose it has $E$ components;
4  Find the shortest path $P_1^*$ between $(m_j, S_1)$;
5  **for** $(i = 1; i \le E-1; i++)$ **do**
6  Find the shortest path $P_{i+1}^*$ between $(S_i, S_{i+1})$
7  Find the shortest path $P_{E+1}^*$ between $(S_E, m_k)$ ;
8  $P^* = P_1^* \cup P_2^* \cup \cdots \cup P_{E+1}^*$ ;
9  **if** *($P^*$ is shorter than $P_d^l$) or ($P_d^l$ is empty)* **then**
10  $P_d^l = P^*$ ;
11 return $P_d$ ;

The idea of the algorithm is to firstly removes the edges $L_s$ from the graph $G$. Suppose the removal of $L_s$ decomposes $G$ into $F$ components, denoted by $G_1, G_2, \cdots, G_F$. We show that we need to add at most $F$ probing paths to prevent $L_s$ from scapegoating attack.

In detail, in subgraph $G_l$, we select a pair of monitors and find the shortest paths $P_1^*$ between the first monitor and one of the segments $S_1$ in $\mathbb{S}$, $P_{E+1}^*$ between the second monitor and another one of the segments $S_E$ in $\mathbb{S}$, and also find the shortest paths $P_2^*, \cdots, P_E^*$ between $(S_1, S_2), \cdots, (S_{E-1}, S_E)$ by using Dijkstra algorithm. The whole path $P_d^l = \{P_1^*, P_2^*, \cdots, P_E^*, P_{E+1}^*\}$ is the defending path of the subgraph $G_l$. Since $L_s$ has been deleted, this path will not traverse any link in $L_s$. The defending paths of the $F$ components cover all segments in $\mathbb{S}$. So at most $F$ defending paths in the $F$ components need to be generated to cover $\mathbb{S}$ and don't traverse any link in $L_S$. The detailed algorithm is given in Algorithm 3. Line 3 to line 8 generate the shortest defending path between two monitors. Line 9 to line 10 select the overall shortest defending path.

The obtained paths pass through all path segments in $\mathbb{S}$ but no link in $L_s$, so they satisfy the requirement to recover the true states of $L_s$. So at most $F$ defending paths are generated to traverse all segments in $\mathbb{S}$ but no links in $L_s$.

The most time consuming step in Algorithm 3 is the Dijkstra algorithm, whose complexity is $O(E * n_l^2)$ where $n_l$ is the number of nodes in $G_l$. Since $n_l < n$, so the complexity of Algorithm 3 is $O\left(F * E * n^2\right)$ in the worst case, where $n$ is the number of nodes in $G$.

An example of defending path generation for $L_s = \{l_1, l_2\}$ is shown in Fig. 5. Fig. 5(a) shows $P_s$ and $L_s$, in which $P_s$ is shown by path segments; Fig. 5(b) shows the result of Algorithm 2, i.e., $\mathbb{S}$, which contains four path segments $\{s_{11}, s_{12}, s_{22}, s_{23}\}$; Fig. 5(c) shows the construction of a defending path by selecting the two monitors, in which blue links

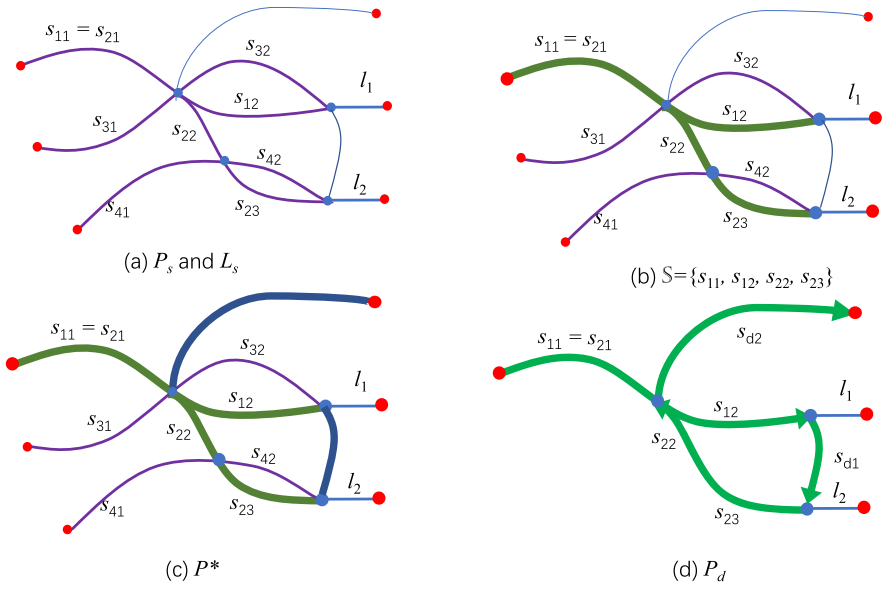



**Fig. 5.** An example of generating the minimum number of defending paths when $L_s$ contains multiple links.

**Table 1**
Parameters of topologies.

| Network | L($\mathcal{G}$) | V($\mathcal{G}$) | Monitors | Paths |
|---|---|---|---|---|
| AttMpls | 56 | 25 | 5 | 100 |
| Surfnet | 68 | 50 | 32 | 845 |
| TataNld | 186 | 145 | 89 | 6217 |
| ER | 1215 | 500 | 61 | 3606 |

show the shortest paths between segment ends, or the shortest path between a monitor and a segment end. Fig. 5(d) showed the constructed shortest defending path $P_d$, which is composed by $\{s_{11}, s_{12}, s_{12}, s_{d1}, s_{23}, s_{d2}\}$.

## 5. Performance evaluation

### 5.1. Experiment setup

We use real network topologies from the Internet Topology Zoo [37] and synthetic ER network topology, whose parameters are shown in the Table 1. Topology Zoo are real ISP network topologies which are widely used in network tomography. For each topology, the MMP [8] algorithm is used to select candidate monitors. After selecting the monitors, the probing paths are generated by multiple shortest paths based on Dijkstra's algorithm and Yen's algorithm [29].

In practice, the difficulty of protecting all links in $L_s$ has relationship with the number of cut sets. Because different selected links in $L_s$ can result in different defense performance, we protect each link separately and calculate average number of cut sets and mean value of defense cost as benchmarks.

### 5.2. Results of defense strategy

Fig. 6 shows the average ratio of identifiable links with the increase of paths in different topologies. The evaluations on both synthetic topology and real network topologies show that when the number of paths is small, the increase of paths has a great influence on the number of identifiable links. But when the number of paths is large to a certain extent, blindly looking for new paths cannot increase the number of identifiable links. The application of our defending path generation algorithm can effectively solve this problem.

We use $|S|/|L(\mathcal{G})|$ to represent the reduction of the size of a graph. $|S|$ is the number of segment in the graph and $|L(\mathcal{G})|$ is the number of links in the graph. The smaller the ratio of $|S|/|L(\mathcal{G})|$, the more significant the size of the graph shrinks. Fig. 7 shows the reduction of the size of the given graphs, achieved by the probing path segmentation method we proposed. Though in the sparse network, the MMP [35] algorithm selects more nodes as candidate monitors, Algorithm 1 still can merge the links and reduce the size of $\mathbb{C}$ efficiently.

Fig. 8 shows an example of how to generate the defending paths in the AttMpls network. The first sub-figure is the network tomography of AttMpls network. The scapegoat link is link 54. There are 10 probing paths, i.e., $P$ in the network,

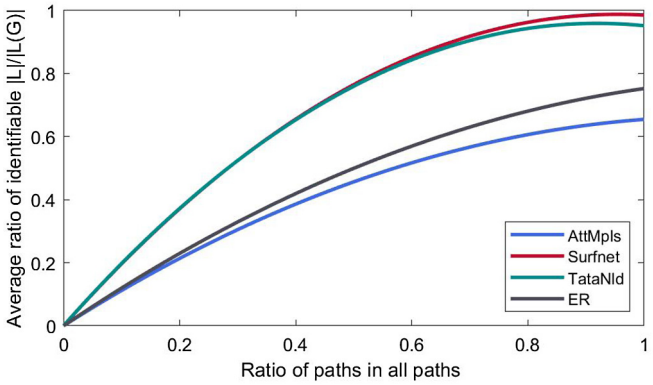



**Fig. 6.** Average ratio of identifiable links with the increase of paths in different topologies.

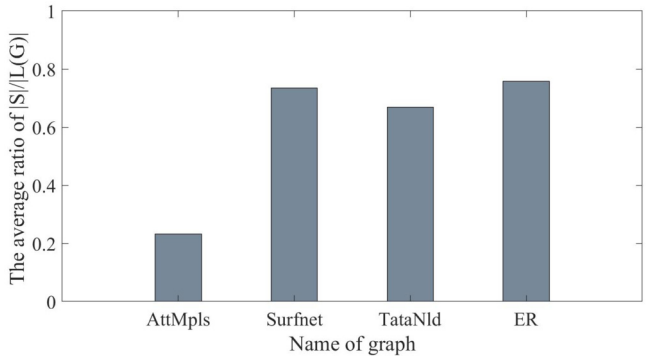



**Fig. 7.** The reduction of the size of graphs achieved by Algorithm 1.

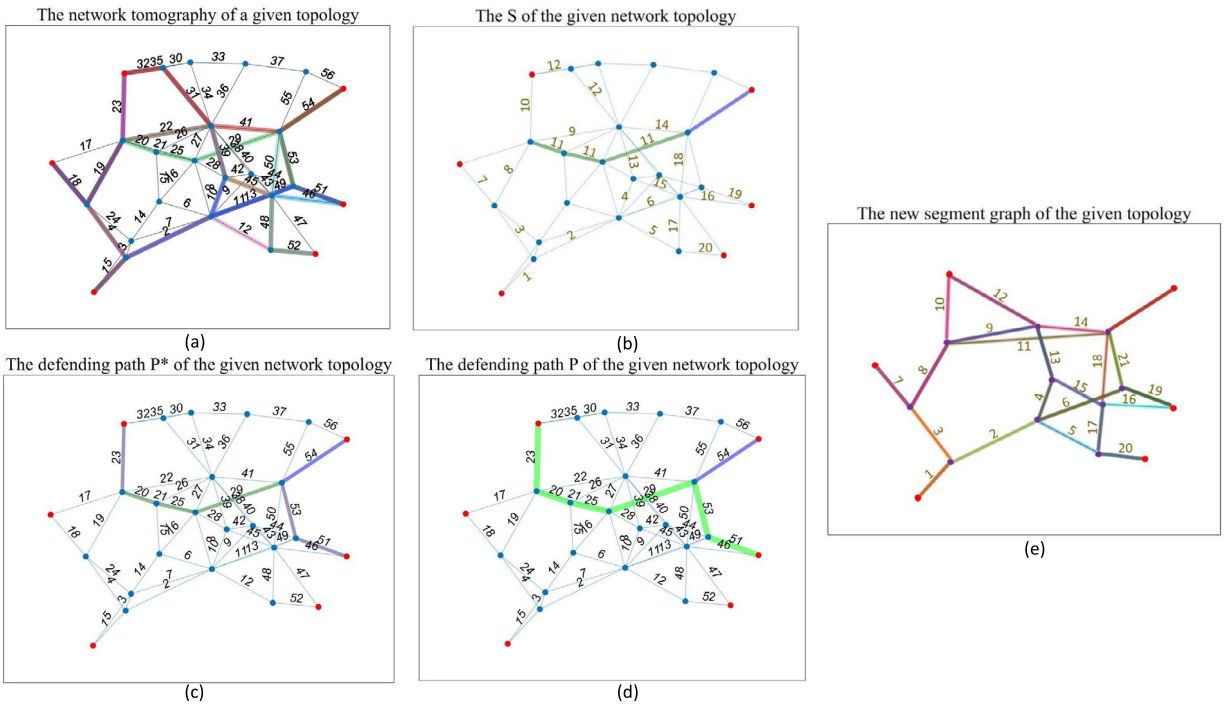



**Fig. 8.** An example of defending path generation.

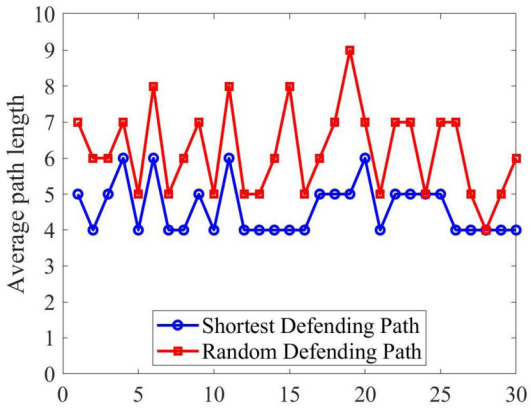



**Fig. 9.** Path length of shortest defense path and random defense path.

which are shown in different colors. Some paths are overlapped so only the top color can be seen. The intersecting vertices of these paths cut each path into several path segments.

To defense, we only consider the paths passing through $L_s$, i.e., $P_s$. We need to insure the minimum segment cut of the segments in $P_s$ being traversed by a defending path.

Firstly, we remove the edges $L_s$ from the graph $G$. According to the defense methodologies, the intersecting vertices cut each path into several path segments. We only need to insure every segment cut having a path segment to be traversed by a defending path. We suppose the possible scapegoat, i.e., $L_s$ is the link 54, which is included in the segment 18.

Based on Algorithm 2, the minimum set cover of the cut set is segment 11, which includes links $\{20, 25, 29\}$. Fig. 8(e) is the new segment graph of the network tomography and Fig. 8(b) is the $\mathbb{S}$ of the given topology, each segment cut in $\mathbb{C}_s$ has at least one segment in $\mathbb{S}$. According to Algorithm 2, the green highlighted links in this figure are the segments $\mathbb{S}$ of the given network tomography. Based on Algorithm 3, the black highlighted links in Fig. 8(c) are the shortest paths $P^*$ between monitors and segments. By using Algorithm 3, we can generate the minimum defending path in Fig. 8(d) to verify whether link 54 is a scapegoat.

We compare two methods of generating defending paths. The first method is randomly choosing a pair of monitors in a subgraph $G_l$ as the source and terminal of the defending path. And the other method is comparing the length of the generated path of all pairs of monitors and find the shortest defending path in the subgraph $G_l$. The second method is more complex than the first one, but the average length of the defending path of the second method is shorter than that of the first method. Fig. 9 is the comparison of two methods, the horizontal x-axis shows the number of experiments in a same subgraph; the vertical y-axis shows the average path length of two methods. As is shown in Fig. 9, when using Algorithm 3, the average path length of finding the shortest defending path is significantly shorter than randomly choose monitors to construct defending path.

## 6. Conclusion

This paper proposes a diagnosis-based defending scheme for scapegoating attack, so that the scapegoating attack can be efficiently detected without the high cost of making all the links in the network be identifiable. We propose a probing path segmentation method to merge the links and reduce the problem of finding link cuts to the problem of finding segment cuts. A minimum set cover model is proposed to select the least number of defense links to cover the unobserved cut set, and we propose a polynomial time algorithm to generate the least number of probing paths to go through the selected defense links.

Theoretical analysis and simulations in the real network topologies show the effectiveness of the proposed defense strategies. In future work, how to optimize the defense strategy proposed in this paper to diagnose and defend against other types of scapegoating attacks to guarantee the security of network tomography should be further studied.

## Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Yongcai Wang reports financial support was provided by National Natural Science Foundation of China, 12071478, 61972404.

## References

[1] X. Xu, Y. Wang, Y. Zhang, D. Li, Defense of scapegoating attack in network tomography, in: The 16th International Conference on Algorithmic Aspects in Information and Management, 2022.

[2] N. Duffield, F.L. Presti, V. Paxson, D. Towsley, Network loss tomography using striped unicast probes, IEEE/ACM Trans. Netw. 14 (2006) 697–710.
[3] A. Chen, J. Cao, T. Bu, Network Tomography: Identifiability and Fourier Domain Estimation, 2007.
[4] T. Bu, N. Duffield, F.L. Presti, D. Towsley, Network tomography on general topologies, in: ACM SIGMETRICS International Conference on Measurement & Modeling of Computer Systems, 2002, p. 21.
[5] Y. Zhao, R. Govindan, D. Estrin, Sensor network tomography: monitoring wireless sensor networks, in: 2001 ACM SIGCOMM Computer Communication Review, vol. 32, 2001.
[6] L. Ma, T. He, K.K. Leung, A. Swami, D. Towsley, Monitor placement for maximal identifiability in network tomography, in: IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, ISSN 0743-166X, 2014, pp. 1447–1455.
[7] S. Tati, S. Silvestri, T. He, T.L. Porta, Robust network tomography in the presence of failures, in: 2014 IEEE 34th International Conference on Distributed Computing Systems, ISSN 1063-6927, 2014, pp. 481–492.
[8] L. Ma, T. He, K. Leung, A. Swami, D. Towsley, Identifiability of link metrics based on end-to-end path measurements, in: 2013 Conference on Internet Measurement Conference, 2013, pp. 391–404.
[9] L. Ma, T. He, K.K. Leung, D. Towsley, A. Swami, Efficient identification of additive link metrics via network tomography, in: Proceedings - 2013 IEEE 33rd International Conference on Distributed Computing Systems, ICDCS 2013, 2013, pp. 581–590.
[10] Y. Qiao, J. Jiao, Y. Rao, H. Ma, Adaptive path selection for link loss inference in network tomography applications, PLoS ONE 11 (2016) e0163706.
[11] S. Zhao, Z. Lu, C. Wang, When seeing isn't believing: on feasibility and detectability of scapegoating in network tomography, in: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), ISSN 1063-6927, 2017, pp. 172–182.
[12] C.-C. Chiu, T. He, Stealthy DGoS attack: DeGrading of service under the watch of network tomography, in: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, IEEE Press, Toronto, ON, Canada, 2020, pp. 367–376.
[13] X. Xu, Y. Wang, L. Xu, D. Li, Locate vulnerable link set to launch minimum cost scapegoating attack in network tomography, under review (2022).
[14] L. Constantin, Attackers slip rogue, backdoored firmware onto Cisco routers | PCWorld, https://www.pcworld.com/article/2984084/attackers-install-highly-persistent-malware-implants-on-cisco-routers.html, 2021.
[15] P. Tague, R. Poovendran, Modeling node capture attacks in wireless sensor networks, in: 2008 46th Annual Allerton Conference on Communication, Control, and Computing, 2008, pp. 1221–1224.
[16] Meng-Fu Shih, A. Hero, Unicast inference of network link delay distributions from edge measurements, in: 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221), vol. 6, ISSN 1520-6149, 2001, pp. 3421–3424.
[17] A. Adams, Tian Bu, T. Friedman, J. Horowitz, D. Towsley, R. Caceres, N. Duffield, F.L. Presti, S.B. Moon, V. Paxson, The use of end-to-end multicast measurements for characterizing internal network behavior, IEEE Commun. Mag. 38 (2000) 152–159.
[18] B. Xi, G. Michailidis, V.N. Nair, Estimating network loss rates using active tomography, J. Am. Stat. Assoc. 101 (2006) 1430–1438.
[19] L. Ma, T. He, A. Swami, D. Towsley, K.K. Leung, J. Lowe, Node failure localization via network tomography, in: Proceedings of the 2014 Conference on Internet Measurement Conference, IMC '14, Association for Computing Machinery, New York, NY, USA, 2014, pp. 195–208.
[20] L. Ma, T. He, A. Swami, D. Towsley, K.K. Leung, Network capability in localizing node failures via end-to-end path measurements, IEEE/ACM Trans. Netw. 25 (2017) 434–450.
[21] N. Bartolini, T. He, V. Arrigoni, A. Massini, F. Trombetti, H. Khamfroush, On fundamental bounds on failure identifiability by Boolean network tomography, IEEE/ACM Trans. Netw. 28 (2020) 588–601.
[22] N. Galesi, F. Ranjbar, Tight bounds for maximal identifiability of failure nodes in Boolean network tomography, in: 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), 2018, pp. 212–222.
[23] Liang Ma, Ting He, Ananthram Swami, Don Towsley, Kin K. Leung, On optimal monitor placement for localizing node failures via network tomography, in: Performance Evaluation, Elsevier Science Publishers B. V. PUB568 Amsterdam, The Netherlands, 2015.
[24] H. Nguyen, P. Thiran, Active measurement for multiple link failures diagnosis in IP networks, vol. 3015, 2004, pp. 185–194.
[25] Y. Xia, D. Tse, Inference of link delay in communication networks, IEEE J. Sel. Areas Commun. 24 (2006) 2235–2248.
[26] A. Gopalan, S. Ramasubramanian, On identifying additive link metrics using linearly independent cycles and paths, IEEE/ACM Trans. Netw. 20 (2012) 906–916.
[27] N. Alon, Y. Emek, M. Feldman, M. Tennenholtz, Economical graph discovery, Oper. Res. 62 (2014) 1236–1246.
[28] T. He, L. Ma, A. Gkelias, K.K. Leung, A. Swami, D. Towsley, Robust monitor placement for network tomography in dynamic networks, in: IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, 2016, pp. 1–9.
[29] T. Pepe, M. Puleri, Network tomography: a novel algorithm for probing path selection, in: 2015 IEEE International Conference on Communications (ICC), 2015.
[30] H.X. Nguyen, P. Thiran, The Boolean solution to the congested IP link location problem: theory and practice, in: IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications, ISSN 0743-166X, 2007, pp. 2117–2125.
[31] A. Yaar, A. Perrig, D. Song, Pi: A Path Identification Mechanism to Defend Against DDoS Attacks, Carnegie Mellon University, 2003.
[32] G. Yao, J. Bi, A.V. Vasilakos, Passive IP traceback: disclosing the locations of IP spoofers from path backscatter, IEEE Trans. Inf. Forensics Secur. 10 (2015) 471–484.
[33] A.T. Mizrak, S. Savage, K. Marzullo, Detecting compromised routers via packet forwarding behavior, IEEE Netw. 22 (2008) 34–39.
[34] J.R. Hughes, T. Aura, M. Bishop, Using Conservation of Flow as a Security Mechanism in Network Protocols, ISSN 1540-7993, IEEE Computer Society, 2000, 0132.
[35] L. Ma, T. He, K.K. Leung, A. Swami, D. Towsley, Inferring link metrics from end-to-end path measurements: identifiability and monitor placement, IEEE/ACM Trans. Netw. 22 (2014) 1351–1368.
[36] V.V. Vazirani, Approximation Algorithms, Springer-Verlag, Berlin, Heidelberg, 2001.
[37] S. Knight, H.X. Nguyen, N. Falkner, R. Bowden, M. Roughan, The Internet topology zoo, IEEE J. Sel. Areas Commun. 29 (2011) 1765–1775.