A Precise Termination Condition of the Probabilistic Packet Marking Algorithm

Tsz-Yeung Wong, Man-Hon Wong, and Chi-Shing (John) Lui, *Senior Member*, *IEEE*

Abstract—The probabilistic packet marking (PPM) algorithm is a promising way to discover the Internet map or an attack graph that the attack packets traversed during a distributed denial-of-service attack. However, the PPM algorithm is not perfect, as its termination condition is not well defined in the literature. More importantly, without a proper termination condition, the attack graph constructed by the PPM algorithm would be wrong. In this work, we provide a precise termination condition for the PPM algorithm and name the new algorithm the *rectified PPM* (RPPM) algorithm. The most significant merit of the RPPM algorithm is that when the algorithm terminates, the algorithm guarantees that the constructed attack graph is correct, with a specified level of confidence. We carry out simulations on the RPPM algorithm and show that the RPPM algorithm can guarantee the correctness of the constructed attack graph under 1) different probabilities that a router marks the attack packets and 2) different structures of the network graph. The RPPM algorithm provides an autonomous way for the original PPM algorithm to determine its termination, and it is a promising means of enhancing the reliability of the PPM algorithm.

Index Terms—Network-level security and protection, probabilistic computation.

1 Introduction

The denial-of-service (DoS) attack has been a pressing problem in recent years [1]. DoS defense research has blossomed into one of the main streams in network security. Various techniques such as the pushback message [2], ICMP traceback [3], and the packet filtering techniques [4], [5], [6], [7] are the results from this active field of research.

The probabilistic packet marking (PPM) algorithm by Savage et al. [8] has attracted the most attention in contributing the idea of IP traceback [9], [10], [11], [12], [13], [14]. The most interesting point of this IP traceback approach is that it allows routers to encode certain information on the attack packets based on a predetermined probability. Upon receiving a sufficient number of marked packets, the victim (or a data collection node) can construct the set of paths that the attack packets traversed and, hence, the victim can obtain the location(s) of the attacker(s).

1.1 The Probabilistic Packet Marking Algorithm

The goal of the PPM algorithm is to obtain a *constructed graph* such that the constructed graph is the same as the *attack graph*, where an attack graph is the set of paths the attack packets traversed, and a constructed graph is a graph returned by the PPM algorithm. To fulfill this goal, Savage et al. [8] suggested a method for encoding the information of the edges of the attack graph into the attack packets through the cooperation of the routers in the attack graph and the victim site. Specifically, the PPM algorithm is made up of two separated procedures: the *packet marking procedure*, which is executed on the router side, and the *graph reconstruction procedure*, which is executed on the victim side.

• *The authors are with the Department of Computer Science and Engineering, the Chinese University of Hong Kong, Ho Sin Hang Engineering Building, Shatin, Hong Kong.*
E-mail: {tywong, mhwong, cslui}@cse.cuhk.edu.hk.

Manuscript received 19 Jan. 2006; revised 10 Dec. 2006; accepted 7 Aug. 2007; published online 6 Sept. 2007.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-0011-0106.
Digital Object Identifier no. 10.1109/TDSC.2007.70229.

The packet marking procedure is designed to randomly encode edges' information on the packets arriving at the routers. Then, by using the information, the victim executes the graph reconstruction procedure to construct the attack graph. We first briefly review the packet marking procedure so that readers can become familiar with how the router marks information on the packets.

1.1.1 A Brief Review of the Packet Marking Procedure

The packet marking procedure aims at encoding every edge of the attack graph, and the routers encode the information in three marking fields of an attack packet: the *start*, the *end*, and the *distance* fields (wherein Savage et al. [8] has discussed the design of the marking fields). In the following, we describe how a packet stores the information about an edge in the attack graph, and the pseudocode of the procedure in [8] is given in Fig. 1 for reference.

When a packet arrives at a router, the router determines how the packet can be processed based on a random number x (line number 1 in the pseudocode). If x is smaller than the predefined marking probability p_m, the router chooses to start encoding an edge. The router sets the start field of the incoming packet to the router's address and resets the distance field of that packet to zero. Then, the router forwards the packet to the next router. When the packet arrives at the next router, the router again chooses if it should start encoding another edge. For example, for this time, the router chooses not to start encoding a new edge. Then, the router will discover that the previous router has started marking an edge, because the distance field of the packet is zero. Eventually, the router sets the end field of the packet to the router's address. Nevertheless, the router increments the distance field of the packet by one so as to indicate the end of the encoding. Now, the start and the end fields together encode an edge of the attack graph. For this encoded edge to be received by the victim, successive routers should choose not to start encoding an edge, that is, the case $x > p_m$ in the pseudocode, because a packet can encode only one edge. Furthermore, every successive router will increment the

Packet Marking Procedure(Packet w)

1. Let x be a random number in $[0 \ldots 1)$
2. **If** $x < p_m$, **then**
3. write router's address into $w.start$ and 0 into $w.distance$
4. **else**
5. **If** $w.distance = 0$ **then**
6. write router's address into $w.end$
7. **end If**
8. increment $w.distance$ by one
9. **end If**

Fig. 1. The pseudocode of the packet marking procedure of the PPM algorithm.

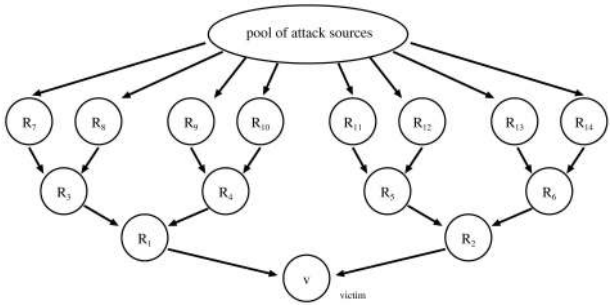

Fig. 2. A 14-router binary-tree network. The upper-bound equation cannot be applied under this multiple-attacker environment.

distance field by one so that the victim will know the distance of the encoded edge.

1.1.2 Termination of the PPM Algorithm

According to the above description of the packet marking procedure, although a packet has already encoded an edge, successive routers may choose to start encoding another edge randomly. As a result, when a packet arrives at the victim, the packet may encode any of the edges of the attack graph, or a packet may not encode any edges. Therefore, if the victim can collect a sufficiently large number of marked packets, the victim can successfully construct all the paths in the attack graph by using the graph reconstruction procedure.

When the graph reconstruction procedure returns a *constructed graph*, it implies the *termination* of the PPM algorithm. However, the *termination condition* has not thoroughly been investigated in the literature. It turns out that the termination condition is important, because it determines the correctness of the constructed graph: If it stops too early, the constructed graph will not contain enough edges of the attack graph and, thus, fails to fulfill the traceback purpose. In addition, it is also not a proper way to allow the victim to collect marked packets for a long period before the victim starts the graph reconstruction procedure, because the victim would never know how much time is long enough. Hence, a proper termination condition can also help in speeding up the traceback process.

In [8], Savage et al. have provided an estimation of the number of marked packets required before the victim can have a constructed graph that is the same as the attack graph under a single-attacker environment. Let X be the number of marked packets required for the victim to reconstruct a path.

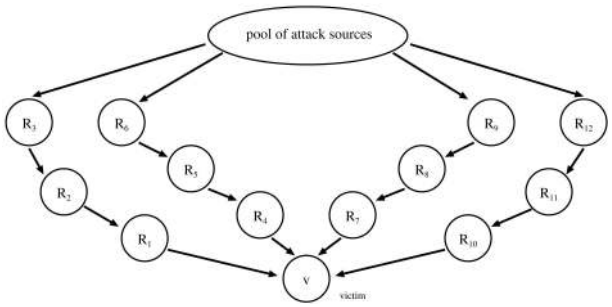

Fig. 3. A 12-router tree network with four independent linear paths, which is another multiple-attacker environment.

Let d be the length of the reconstructed path. In addition, let p_m be the marking probability of every router in the path. The upper-bound on the expectation $E[X]$ is given in [8, Equation (1)], and we name this equation the *upper-bound equation* throughout this paper

$$E[X] < \frac{\ln(d)}{p_m(1-p_m)^{d-1}}. \quad (1)$$

1.2 Problems When Using the Upper-Bound Equation as the Termination Condition

Although there is no explicit definition of the termination condition of the PPM algorithm in [8], it is well accepted that (1) is the termination condition in the single-attack environment. The authors also claimed that in a multiple-attacker environment

> *The number of packets needed to reconstruct each path is independent, so the number of packets needed to reconstruct all paths is a linear function of the number of attackers.*

However, we have found that this is not the case in general. More specifically, (1) should not be treated as the termination condition of the PPM algorithm.

1.2.1 Failure in the Multiple-Attacker Environment

First, one cannot apply the termination condition to complex networks such that the reconstruction of one path is dependent on another. This scenario can be explained in Fig. 2, which is a binary-tree network with 14 routers. The leaf routers from R_7 to R_{14} are connected to a pool of attackers. These attackers send out attack traffic toward the victim v, and this presents a multiple-attacker environment. In this graph, the attack packets traversed through eight paths that are identical in structure. However, there are "shared" edges among these paths. This implies that the reconstruction of one path is dependent on another. Therefore, one cannot treat (1) as the termination condition under this scenario, and this restricts the application of the PPM algorithm.

Second, although every path in a given network is independent, we have found that the number of marked packets needed to reconstruct the network graph does not have a linear relationship with the number of paths; that is, the claim made in [8] is not correct. We have carried out a set of simulations to show our finding and we start the description of our simulation setup from the network depicted in Fig. 3. The network contains four paths that are identical in structure and, more importantly, there are no shared edges between any two paths. We name these paths the *independent paths*. In addition, we assume that one independent path connects to

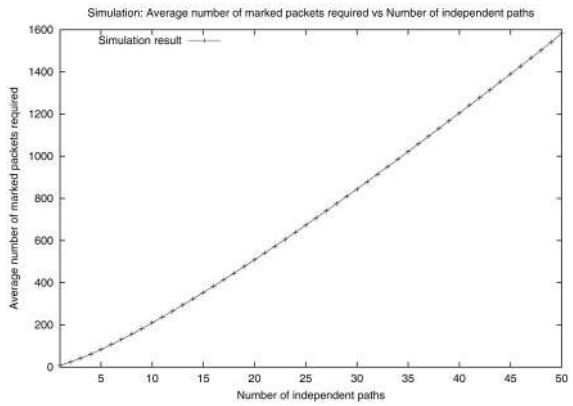

Fig. 4. The relationship between the number of independent paths and the average number of marked packets required.

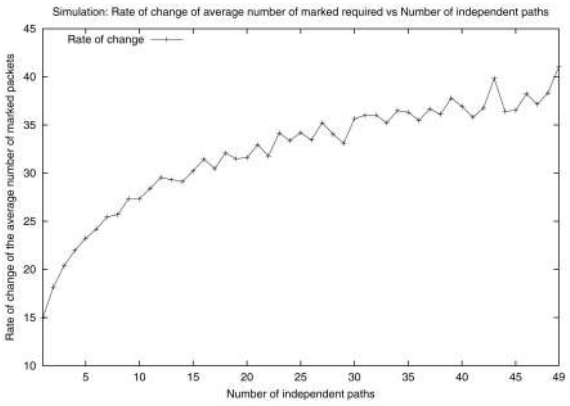

Fig. 5. An increasing trend in the rate of change in the number of marked packets required.

one attacker and every attacker sends out a similar amount of attack traffic toward the victim.

We then carry out a simulation to obtain the average number of marked packets required to reconstruct the paths. Next, we repeat this simulation, but this time, we add one more independent path to the network, and there are now five independent paths. Eventually, we perform a series of simulations for one to 50 independent paths. Fig. 4 shows the result of this set of simulations. One can observe that the average number of marked packets required to construct a correct constructed graph increases as the number of independent paths increases. In order to show whether the number of required marked packets linearly increases with the number of paths or not, we plot the rate of change in the number of required marked packets in Fig. 5. Surprisingly, the graph shows an increasing trend in the rate of change in the number of required marked packets. The claim about the multiple-attacker environment made in [8] is therefore wrong.

Theoretically, the packet collecting problem can be transformed into the "*coupon-collecting problem with unequal probabilities*" [15]. The fault made in [8] is to treat the probability that every encoded edge arrived at the victim the same, which is wrong (we will discuss this in Section 3). The solution to the coupon-collection problem with unequal probabilities is very complex and does not show a linear property with the number of the independent paths.

In summary, the first problem of using (1) as the termination condition is that the relationship between the number of attack paths and $E[X]$ is not known. Therefore, the PPM algorithm cannot guarantee the correctness under the multiple-attacker environment.

1.2.2 Another Problem

No matter how accurate the calculation of the expectation $E[X]$ is, one should not use the expected number of required marked packets $E[X]$ as the termination condition. Depending on the underlying probability distribution of the random variable X, when the mean is reached, there is a nonzero probability that the constructed graph is still an incorrect one. For instance, if the probability distribution of X is a uniform distribution, then the probability that a correct attack graph is constructed is just 0.5. In summary, when X has high variance, the first moment estimation may not be accurate.

Based on the above two problems, we conclude that the upper-bound equation is not suitable to be the termination condition of the PPM algorithm.

1.3 Contributions and Paper Structure

In this work, we neither provide an accurate calculation of $E[X]$ nor discover the probability distribution of the random variable X. Instead, we modify the PPM algorithm so that the victim can obtain a correct constructed graph with a *specified level of guarantee*. The contributions of this work are listed as follows:

- We introduce the *termination condition* of the PPM algorithm, which is missing or is not explicitly defined in the literature.
- Through the new termination condition, the user of the new algorithm is free to determine the correctness of the constructed graph.
- The constructed graph is guaranteed to reach the correctness assigned by the user, independent of the marking probability and the structure of the underlying network graph.

The structure of this paper is organized as follows: Section 2 describes the modifications of the PPM algorithm, and we name the new algorithm the *rectified PPM* (RPPM) algorithm. In turn, the termination condition of the RPPM algorithm is again expressed in terms of the number of collected marked packets, but the number changes based on the size of the constructed graph. We name that number the *termination packet number* (TPN). Before deriving the calculation of the TPN, we present the modeling of the packet marking procedure in Section 3. In Section 4, we derive the calculation of the TPN. Section 5 provides the simulation results, which show the correctness and the robustness of the RPPM algorithm. Section 6 discusses how the RPPM algorithm adopts the relaxation of the assumptions made in Section 2. In Section 7, we discuss some deployment issues of the RPPM algorithm. Last, Section 8 concludes.

2 Rectified Probabilistic Packet Marking Algorithm

The *RPPM* algorithm is designed to automatically determine when the algorithm should terminate. We aim at achieving the following properties:

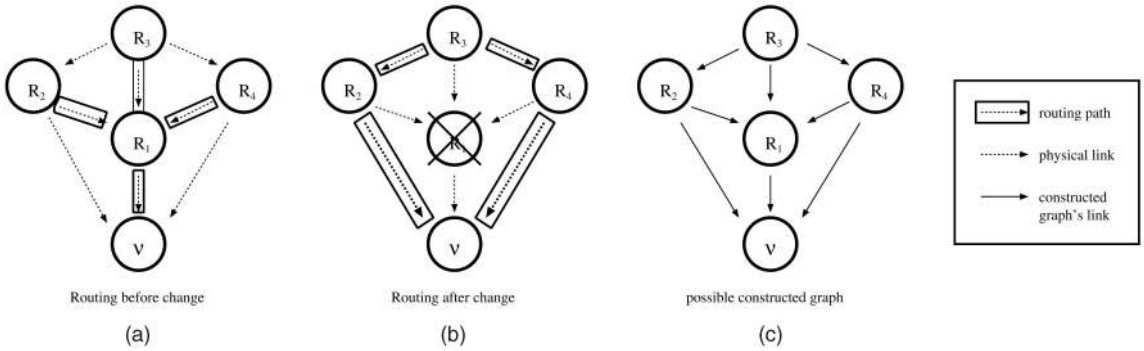

Fig. 6. The failure of the router R_1 causes the route tables of R_2, R_3, and R_4 to change. This results in a constructed graph with routers that have multiple outgoing edges.

1. The algorithm does not require any prior knowledge about the network topology.
2. The algorithm determines the certainty that the constructed graph is the attack graph when the algorithm terminates.

Our goal is to devise an algorithm that guarantees that the constructed graph is the same as the attack graph with probability greater than P^*, where we name P^* the *traceback confidence level* (it is analogous to the level of confidence that the algorithm wants to achieve). To accomplish this goal, the graph reconstruction procedure of the original PPM algorithm is completely replaced, and we name the new procedure the *rectified graph reconstruction procedure*. On the other hand, we preserve the packet marking procedure so that every router deployed with the PPM algorithm is not required to change.

In the following section, we list the assumptions of our solution. Then, we describe the flow of the *rectified graph reconstruction procedure*.

2.1 Assumptions

2.1.1 Assumptions about the Router

For each router, we assume that it is equipped with the ability to mark packets as in the original PPM algorithm. We also assume that each router shares the same marking probability. Specifically, a router can either be a transit router or a leaf router. A transit router is a router that forwards traffic from upstream routers to its downstream routers (or the victim), whereas a leaf router is a router whose upstream router is connected to client computers (not routers) and forwards the clients' traffic to its downstream routers (or the victim). Certainly, the clients are mixed with honest and malicious parties. In addition, we assume that all leaf routers in an attack graph are the sources of the attack packets, and each leaf router sends out a similar number of attack packets. Note that we are not assuming that there is only one attacker, but we are considering a multiple-attacker environment.

Furthermore, we assume that every router has only one outgoing route toward the victim. For the ease of presentation, we name the "outgoing route toward the victim" the *victim route*. The assumption can be justified by the fact that modern routing algorithms favor the construction of routing trees [16], [17]. This assumption is also reflected in the structures of the constructed graph: every router in the constructed graph has only one outgoing edge. However, this assumption may not hold under abnormal situations. For example, in Fig. 6, the failure of the router R_1 forces the routing table to completely change. Under such a scenario, the constructed attack graph may become the one shown in Fig. 6c. We argue that this result is not an undesirable one, as long as the definition of a correct attack graph construction still holds (because the new attack graph is indeed composed of all the edges traversed by the packets). In the remainder of this paper, we stay with this assumption, and we will discuss the scenario when this assumption is relaxed in Section 6.

2.1.2 Assumptions about the Victim

On the victim side, we assume that by the time that the victim starts collecting marked packets, all routers in the network have already invoked the packet marking procedure. In addition, we assume that the victim does not have any knowledge about the real network or the attack graph. However, the victim knows the marking probability that the routers are using.

2.2 Flow of the Rectified Graph Reconstruction Procedure

The pseudocode of the rectified graph reconstruction procedure is shown in Fig. 7, and the procedure is started as soon as the victim starts collecting marked packets. When a marked packet arrives at the victim, the procedure first checks if this packet encodes a new edge. If so, the

Rectified Graph Reconstruction Procedure (Traceback Confidence Level P^*)

```
/* Initially, G_e contains the "victim" node only, and pkt_count = 0. */
1. Foreach incoming packet pkt ; do
2.    pkt_count := pkt_count + 1;
3.    If the incoming packet pkt contains an edge e that is not included in G_e; then
4.          Construct the new attack graph G_e by inserting the edge e ;
5.          If G_e is a connected graph ; then
6.              TPN := TPN_subroutine(G_e, P*) ;
7.              pkt_count := 0 ;
8.          end If
9.    end If
10.   If G_e is a connected graph ; then
11.         If pkt_count > TPN ; then
12.             Return G_e as the constructed graph ;
13.         end If
14.   end If
15. end Foreach
```

Fig. 7. The pseudocode of the rectified graph reconstruction procedure of the RPPM algorithm.

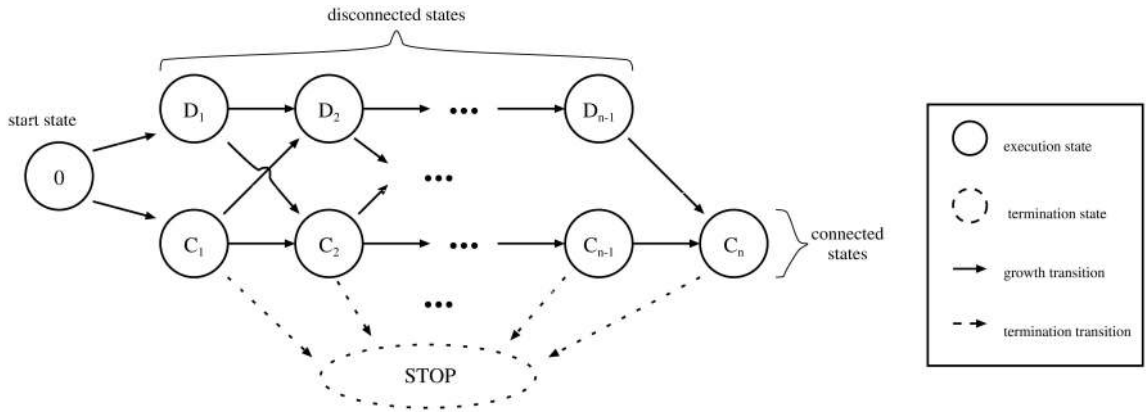

Fig. 8. An execution diagram of the rectified graph reconstruction procedure of the RPPM algorithm that constructs a graph with n edges.

procedure accordingly updates the constructed graph G_c. Next, if the constructed graph is *connected*, where *connected* means that every router can reach the victim, the procedure calculates the number of incoming packets required before the algorithm stops, and we name this number the *TPN*. The procedure then resets the counter for the incoming packets to zero and starts counting the number of incoming packets. In the meantime, the procedure checks if the number of collected packets is larger than the TPN. If so, the procedure claims that the constructed graph G_c is the attack graph, with probability P^*. Otherwise, the victim receives a packet that encodes a new edge. Then, the procedure updates the constructed graph, revisits the TPN calculation subroutine, resets the counter for incoming packets, and waits until a packet that encodes a new edge arrives or the number of incoming packets is larger than the new TPN.

As suggested by the pseudocode, the termination condition of the RPPM algorithm is that "*the counter for the incoming packets is larger than the TPN*," and this implies that the calculation of the TPN during each update of the constructed graph is the core of the RPPM algorithm. In the next step, we provide a deeper understanding of the RPPM algorithm through the introduction of the *execution diagram*.

2.3 Execution Diagram of the Rectified Probabilistic Packet Marking Algorithm

According to the previous section, it is observed that the TPN, the constructed graph, and the execution of the rectified graph reconstruction procedure are closely related. Such a relationship can be visualized by the construction of the *execution diagram*, as shown in Fig. 8. The execution diagram presents the dynamics of the execution of the rectified graph reconstruction procedure.

2.3.1 Types of States

There are two types of states in the diagram: the *execution state* and the *termination state*. When the procedure is running, we say that "*the rectified graph reconstruction procedure is in an execution state*." Otherwise, we say that "*the rectified graph reconstruction procedure is in the termination state*." The execution state also tells us the state of the constructed graph: 1) when the procedure is in the *start state*, labeled by "0," it means that the procedure has started running, and there are no edges in the constructed graph. 2) When the procedure is in a *connected state*, it means that the constructed graph is connected. A connected state, labeled by C_i, means that the constructed graph is connected and contains i edges. 3) When the procedure is in a *disconnected state*, the constructed graph is disconnected. A disconnected state, labeled by D_i, means that the constructed graph is disconnected and contains i edges. Note that both the connected and disconnected states, say, C_i and D_i, respectively, refer to all the possible graphs that have i edges. Last, when the procedure is in the *termination state*, it means that the procedure has stopped.

2.3.2 Types of Transitions

There are two kinds of transitions in the execution diagram. When the procedure takes a *growth transition*, it means that a new edge is added to the constructed graph. When the procedure takes a *termination transition*, it means that the procedure is going to stop running.

The transition structure in Fig. 8 is derived from the pseudocode of the rectified graph reconstruction procedure in Fig. 7. We briefly describe the transition structure as follows: 1) If a packet that encodes a new edge arrives before the number of received packets is larger than the TPN, then the procedure takes a growth transition and proceeds to either a connected state or a disconnected state, depending on the connectivity of the updated constructed graph. 2) If the number of received packets is larger than the TPN, then the procedure takes the termination transition and proceeds to the termination state. 3) If the procedure is in one of the disconnected states, then it is meaningless to return such a graph as the correct constructed graph, and there is no transition that connects the disconnected states to the termination state. The procedure then keeps on collecting packets until it proceeds to a connected state.

2.3.3 Worst-Case, Average-Case, and Best-Case Scenarios

According to the execution diagram, one can classify three kinds of execution scenarios of the RPPM algorithm. They are the *worst-case*, the *average-case*, and the *best-case* scenarios. This classification is based on the possibility that the RPPM algorithm returns a correct graph.

If one assumes that the constructed graph is always connected, then at every state, the victim has to calculate the TPN and has to wait until the rectified graph reconstruction procedure makes a transition to the next connected state or the termination state. In other words, the procedure is *vulnerable*, returning an incorrect result, because there is always a nonzero probability that the procedure is terminated. We name this scenario the *worst-case scenario*. On the other hand, if the constructed graph is allowed to

enter a disconnected state, then the procedure would not always have the possibility of entering the termination state. We name this scenario the *average-case scenario*.

In addition, there is a possibility that the rectified graph reconstruction procedure is always in the disconnected states (except for the state when the constructed graph becomes the attack graph). Then, there is no chance for the procedure to return an incorrect result. We name this scenario the *best-case scenario*. Note that the best-case scenario will always have a successful graph reconstruction.

2.4 Role of the Execution Diagram

The execution diagram provides a thorough understanding of the relationship among the execution of the rectified graph reconstruction procedure, the constructed graph, and the TPN. Through the analysis of the execution diagram, it can be observed that different execution scenarios of the procedure would affect the probability that the procedure returns a correct constructed graph.

It is observed that the worst-case scenario would be the hardest case for the rectified graph reconstruction procedure to return a correct graph. Therefore, it is an ideal point for us to derive the calculation of the TPN. Supposing that one could successfully provide a guarantee of the correctness of the constructed graph under the worst-case scenario, then such a guarantee can also be provided in the average-case scenario. Moreover, it is expected that the average-case scenario should outperform the worst-case scenario in terms of the successful rate of returning a correct constructed graph. Next, we will move on to the modeling of the packet marking process of the packet marking procedure.

3 Packet-Type Probability

As defined in Section 1.1.1, the packet marking procedure is the source of different kinds of marked packets, and the total number of possible marked packets is the number of edges of the attack graph. However, it will be shown in the next section that the probability for every kind of marked packets that arrive at the victim plays a vital part in the derivation of the termination packet number. In this section, we present the definition and the derivation of such a set of probabilities, and we name them the *packet-type probabilities*.

3.1 Encoded Edge Random Variable

By definition, an incoming packet may encode one of the edges of the attack graph, or the incoming packet does not encode any edges of the attack graph. We use a random variable called the *encoded edge* random variable to represent all possible encodings on an incoming packet. We formally define the encoded edge random variable as follows:

Definition 1. *Define $T(G)$ as the encoded edge random variable. $T(G) = e$ represents that a packet encoding the edge e arrives at the victim, where e is in the set of edges of the attack graph G. In addition, define $T(G) = \phi$ if the packet that arrived at the victim does not encode any edge.*

For each value of the encoded edge random variable, there is a corresponding probability for that value and it is called the *packet-type probability*.

3.2 Calculating the Packet-Type Probability

Let the attack graph be $G = (V, E)$. In addition, let $R_i, R_j \in V$ and $(R_i, R_j) \in E$. Suppose that we are interested in the probability that a packet encodes the edge (R_i, R_j). Without loss of generality, the proposed solution can also deal with the edges in the form (R_i, v), where v is the victim site. To begin with, the packet-type probability $P(T(G) = (R_i, R_j))$ can be expressed as

$$\begin{aligned} P(T(G) = (R_i, R_j)) &= P(\text{"a packet passes through } (R_i, R_j)\text{"} \\ &\qquad \text{and "a packet encodes } (R_i, R_j)\text{"}). \\ &= P(\text{"a packet passes through } (R_i, R_j)\text{"}) \\ &\quad \times P(\text{"a packet encodes } (R_i, R_j)\text{"} \\ &\quad \mid \text{"a packet passes through } (R_i, R_j)\text{"}). \end{aligned}$$

For the ease of presentation, we name the probability P("a packet passes through (R_i, R_j)") the *via probability*. In addition, we name the probability P("a packet encodes (R_i, R_j)" | "a packet passes through (R_i, R_j)") the *conditional encoding probability*.

3.2.1 Via Probability

Let $L(G)$ be the set of leaf routers in G and let $|L(G)|$ be the number of leaf routers in $L(G)$. In addition, let $Path(R, v)$ be the set of paths that lead from the router R to the victim v and let $|Path(R, v)|$ be the number of paths in $Path(R, v)$. Moreover, we assume that every path will have an equal chance to be chosen by a packet.

Let R_l be a leaf router in G. If there is only one path in the set $Path(R_l, v)$ that contains (R_i, R_j), then the via probability under this specific case is given by

$$\textbf{Via probability } (\text{single-path case}) = \frac{1}{|L(G)|} \times \frac{1}{|Path(R_l, v)|}. \tag{2}$$

Furthermore, because the event that a packet passes through one path is independent of the event that a packet passes through another path, if there is more than one path that contains the edge (R_i, R_j), the probability that a packet passed through (R_i, R_j) will be the sum of a collection of the probabilities for the single path cases in (2). Let $\delta(r, (R_i, R_j))$ be a function such that if the path r contains the edge (R_i, R_j), then it returns one; otherwise, it returns zero. Then, the via probability is given as follows:

$$\begin{aligned} \textbf{Via probability} = &\sum_{R_l \in L(G)} \sum_{r \in Path(R_l, v)} \delta(r, (R_i, R_j)) \\ &\times \frac{1}{|L(G)|} \times \frac{1}{|Path(R_l, v)|}. \end{aligned} \tag{3}$$

3.2.2 The Conditional Encoding Probability

The conditional encoding probability is concerned with how the packet's markings can reach the victim without being overwritten. The formulation of this probability relies on the distance between the edge and the victim. We call the distance function the *edge distance function*, and it is given by

$$d((R_i, R_j), v, r) = \begin{cases} 1, & R_j = v, \\ d\big((R_j, R_k), v, r\big) + 1, & \text{otherwise}, \end{cases} \tag{4}$$

where R_k is one hop closer to the victim than R_j on the path r.

For every path that contains the edge (R_i, R_j), if a packet encodes the edge (R_i, R_j), then it means that R_i marked the start field of the packet, whereas successive routers on that path did not mark the start field. Then, the conditional

```
Packet_type(Graph G)

/* Number of elements in the "result" array is equal to the number of edges of graph "G". */
1. result := allocate_memory(G.edge) ;
2. For (i := 0; i < G.edge; i := i + 1)
3.     result[i] := 0 ;
4. end For
5. Foreach leaf in G ; do
/* "search_path" finds all the paths from leaf to victim */
6.     path_set := search_path(leaf, victim) ;
7.     Foreach path in path_set ; do
8.         Foreach edge in path ; do
9.             length := edge_distance_function(edge, victim, path) ;
10.            result[edge] := result[edge] + 1.0/G.leaf_num × 1.0/path.num ×
11.                                p_m × (1 − p_m)^(length−1);
12.        end Foreach
13.    end Foreach
14. end Foreach
15. return result ;
```

Fig. 9. The pseudocode of the packet-type probability calculation subroutine. It calculates the packet-type probability of every edge of the input graph, specified by G.

encoding probability, given that the incoming packet follows the path r, is

$$\begin{aligned}&\textbf{Conditional encoding probability}(\text{on path } r)\\&\quad = p_m(1-p_m)^{d((R_i,R_j),v,r)-1}.\end{aligned}$$

Finally, we have the packet-type probability of (R_i, R_j) as follows:

$$\begin{aligned}&P(T(G)=(R_i,R_j))\\&\quad=\sum_{R_l\in L(G)}\sum_{r\in Path(R_l,v)}\delta(r,(R_i,R_j))\times\frac{1}{|L(G)|}\\&\qquad\times\frac{1}{|Path(R_l,v)|}\times p_m\times(1-p_m)^{d((R_i,R_j),v,r)-1}.\end{aligned}\tag{5}$$

In addition, the packet-type probability of an unmarked packet is given as follows:

$$P(T(G)=\phi)=1-\sum_{e\in E}P(T(G)=e),\tag{6}$$

where E is the edge set of $G=(V,E)$.

Note that the above derivation of the packet-type probability includes the presence of the unmarked packets. If the victim considers only marked packets, a suitable normalization should be applied as follows: Denote $T_m(G)$ as the *strict encoded edge* random variable, which is the same as the encoded edge random variable $T(G)$, except that $T_m(G)$ takes on only values of the edge set E of the graph G, that is, without the value ϕ. Then, the *strict packet-type probability* is given as follows:

$$P(T_m(G)=e)=\frac{P(T(G)=e)}{1-P(T(G)=\phi)},\ \forall e\in E.\tag{7}$$

3.2.3 The Pseudocode of the Calculation of the Packet-Type Probabilities

In Fig. 9, we provide an algorithm for calculating the packet-type probability of every edge of an input graph. The algorithm first constructs the paths that lead from every leaf router to the victim. Then, for each path, the algorithm calculates and accumulates the packet-type probability by (5) for every edge in the path. Eventually, it returns the packet-type probabilities of all edges of the input graph. Note that the calculations of the packet-type probability for an unmarked packet and the strict packet-type probabilities are not included in the pseudocode, but one can calculate these probabilities by using (6) and (7), together with the results obtained by the algorithm.

After deriving the calculation of the packet-type probability, we are ready for the calculation of the termination packet number. In the next section, we derive the calculation of the termination packet number.

4 Derivation of the Termination Packet Number

In this section, we present the calculation of the TPN at each connected state (see Section 2.3) so that the RPPM algorithm returns a correct constructed graph, with probability larger than P^*. As mentioned at the end of Section 2, we assume that the constructed graph is always connected; that is, we consider only the worst-case case scenario.

We denote $P_{\tau_i}(C_i \to C_{i+1})$ as the probability that the rectified graph reconstruction procedure proceeds from state C_i to state C_{i+1}, with the TPN set to τ_i, and we name this probability the *state-change probability from C_i to C_{i+1}*. In other words, it is the probability that the victim receives a new edge before the number of collected marked packets is larger than the TPN τ_i. Note that we are not referring to any specific

constructed graphs. Instead, as mentioned in Section 2.3.1, C_i represents all the possible connected graphs with i edges.

Since the probability that the RPPM algorithm that returns a correct constructed graph is equivalent to the probability that the RPPM algorithm makes a transition of $n-1$ steps from states C_1 to C_n, mathematically, we have the following:

$$P(\text{constructed graph is correct}) = \prod_{j=1}^{n-1} P_{\tau_j}(C_j \rightarrow C_{j+1}).$$

Then, our claim is correct, given that the product of the state-change probabilities from states C_1 to C_n should be greater than P^* and is given by

$$\prod_{j=1}^{i} P_{\tau_j}(C_j \rightarrow C_{j+1}) > P^*.$$

For the sake of further presentation, we transform the above equation as follows:

$$P_{\tau_i}(C_i \rightarrow C_{i+1}) > \frac{P^*}{X_{i-1}}, \quad \text{where } X_{i-1} = \prod_{j=1}^{i-1} P_{\tau_j}(C_j \rightarrow C_{j+1}). \tag{8}$$

Note that X_{i-1} in (8) is the product of the state-change probabilities of the past states of the rectified graph reconstruction procedure, and we named it the *accumulated state-change probability at state* C_i. We will discuss how we can calculate the accumulated state-change probability in Section 4.1.4.

4.1 Termination Packet Number Derivation

According to the previous section, we know that the TPN at each connected state can be found by (8), which is expressed in terms of the state-change probability. In this section, we derive the TPN by deriving the state-change probability with the following steps:

1. To recall, the state-change probability is the probability that the constructed graph of state C_i evolves into the constructed graph of state C_{i+1}. Hence, the first step in calculating the state-change probability is to find all the graphs that could possibly be the next constructed graph, and we name this set of graphs the *extended graphs*.
2. In the second step, for each extended graph G_e, we find the probability that the current constructed graph becomes the extended graph G_e. As a matter of fact, the above probability is the state-change probability from C_i to C_{i+1}, *conditioned that* the extended graph G_e is the next constructed graph, and we name this the *conditional state-change probability*.
3. From the conditional state-change probability, one can find the state-change probability (and, thus, the TPN) through the definition of the condition probability. Nevertheless, because the calculation of the exact TPN violates the basic assumptions of the traceback problem, the *upper-bounded* TPN would alternatively be derived, and the relationship between the exact TPN and the upper-bounded TPN will be presented.

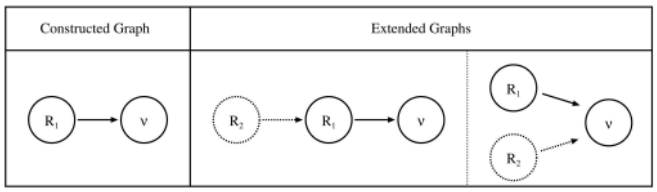

Fig. 10. An illustration of the concept of the extended graph.

4.1.1 Extended Graphs

The extended graphs are the predictions of the future constructed graph based on the current graph. Denote the constructed graph in state C_i of the rectified graph reconstruction procedure as G_i, where $i \geq 1$. By the assumption that every router has only one victim route (stated in Section 2.1) and the assumption that every constructed graph is connected (which was made earlier in this section), when the constructed graph evolves from G_i to G_{i+1}, there are always one new edge and one new node inserted into G_i.

The example in Fig. 10 helps illustrate the above point. On the left side of the figure, there is a constructed graph with one edge that connects two nodes, and the victim and the router are labeled by v and R_1, respectively. On the right side of the figure, a new edge is inserted in the constructed graph at two possible locations: the graph on the left has the new edge (R_2, R_1), and another one has the new edge (R_2, v). We name the introduced edges the *extended edges*. Formally, we define the extended graphs of G_i in Definition 2, and we define $\mathcal{G}(G_i)$ as the set of extended graphs.

Definition 2. *Let $\mathcal{G}(G_i)$ be the set of extended graphs of the constructed graph $G_i = (V_i, E_i)$ in state C_i of the rectified graph reconstruction procedure:*

$$\mathcal{G}(G_i) = \{G_e = (V_e, E_e) \mid \exists (u,t) \notin E_i \ \& \ u \notin V_i \ \& \ t \in V_i \text{ such that } V_e = V_i \cup u \text{ and } E_e = E_i \cup (u,t)\}.$$

By the assumption that every constructed graph is connected in this section, $\mathcal{G}(G_i)$ has already included all the possible candidates for the next constructed graph G_{i+1}. Thus, in the next step, we assume that an extended graph G_e is the next constructed graph G_{i+1}. Then, we calculate the state-change probability, conditioned that $G_{i+1} = G_e$, and we call it the *conditional state-change probability*. Last, by using the definition of conditional probability

$$P_{\tau_i}(C_i \rightarrow C_{i+1}) = \sum_{G_e \in \mathcal{G}(G_i)} P_{\tau_i}(C_i \rightarrow C_{i+1} \mid G_{i+1} = G_e) \times P(G_{i+1} = G_e),$$

we have the state-change probability.

4.1.2 The Conditional State-Change Probability

The conditional state-change probability is calculated according to the following rationale. If one assumes that $G_{i+1} = G_e$, then one knows the topology of the next constructed graph and also knows where the extended edge is. Then, the state-change probability is equivalent to the probability that a packet that encodes the extended edge arrives at the victim before the number of collected packets is larger than the TPN.

The probability that the extended edge e' arrives at the victim is exactly the packet-type probability $P(T(G_e) = e')$.

Because the marking process of each packet is independent, the state-change probability, conditioned that $G_{i+1} = G_e$, is therefore given by the following:

$$P_{\tau_i}(C_i \to C_{i+1} \mid G_{i+1} = G_e) = 1 - \Big(1 - P(T(G_e) = e')\Big)^{\tau_i}. \quad (9)$$

Note that (9) is an increasing function with respect to τ_i, because

$$\frac{d}{dx}\left(1 - \Big(1 - P(T(G_e) = e')\Big)^{x}\right) = -\Big(1 - P(T(G_e) = e')\Big)^{x} \log\Big(1 - P(T(G_e) = e')\Big) > 0,$$

where $x > 0$ & $P(T(G_e) = e') \in (0, 1)$.

To continue with the calculation of the state-change probability, the probability $P(G_{i+1} = G_e)$ has to be known. However, this is prohibited by the assumption that the victim does not have any information about the attack graph. As an alternative, the *upper-bounded TPN* will be derived instead.

4.1.3 Upper-Bounded TPN

Since the conditional state-change probability increases with respect to τ_i (stated in the note of (9)), one can always find a sufficiently large integer τ_i^* such that

$$P_{\tau_i^*}(C_i \to C_{i+1} \mid G_{i+1} = G_e) > \frac{P^*}{X_{i-1}},\ \forall\, G_e \in \mathcal{G}(G_i). \quad (10)$$

By the above idea, we have

$$\begin{aligned} P_{\tau_i^*}(C_i \to C_{i+1}) &= \sum_{G_e \in \mathcal{G}(G_i)} P_{\tau_i^*}(C_i \to C_{i+1} \mid G_{i+1} = G_e) \times P(G_{i+1} = G_e) \\ &> \sum_{G_e \in \mathcal{G}(G_i)} \frac{P^*}{X_{i-1}} \times P(G_{i+1} = G_e). \qquad \text{(by (10))} \end{aligned}$$

$$\because \sum_{G_e \in \mathcal{G}(G_i)} P(G_{i+1} = G_e) = 1, \quad \therefore P_{\tau_i^*}(C_i \to C_{i+1}) > \frac{P^*}{X_{i-1}}.$$

Hence, this shows that τ_i^* can also be a TPN of state C_i, because (8) is satisfied. By the above arguments, it is required to confirm the existence of τ^* such that τ_i^* is large enough to satisfy (10). From (10), we have

$$\begin{aligned} & P_{\tau_i^*}(C_i \to C_{i+1} \mid G_{i+1} = G_e) > \frac{P^*}{X_{i-1}} \\ \Rightarrow\ & 1 - \Big(1 - P(T(G_e) = e')\Big)^{\tau_i^*} > \frac{P^*}{X_{i-1}} \qquad \text{(by (9))} \\ \Rightarrow\ & \tau_i^* > \frac{\log\left(1 - \frac{P^*}{X_{i-1}}\right)}{\log\big(1 - P(T(G_e) = e')\big)}. \end{aligned}$$

Since the TPN is an integer, we have

$$\tau_i^* = \lfloor Y_i(G_e) + 1 \rfloor, \text{ where } Y_i(G_e) = \frac{\log\left(1 - \frac{P^*}{X_{i-1}}\right)}{\log\big(1 - P(T(G_e) = e')\big)}.$$

Furthermore, by the monotonic increasing property of the logarithmic function, $Y_i(G_e)$ is monotonic decreasing with respect to $P(T(G_e) = e')$. Thus, by finding the value $\min_{G_e \in \mathcal{G}(G_i)} P(T(G_e) = e')$, the maximum value of τ_i^* in the set of extended graphs $\mathcal{G}(G_i)$ can be found. Therefore,

$$\tau_i^* = \left\lfloor \frac{\log\left(1 - \frac{P^*}{X_{i-1}}\right)}{\log(1 - p_{min})} + 1 \right\rfloor, \text{ where } p_{min} = \min_{G_e \in \mathcal{G}(G_i)} P(T(G_e) = e'). \quad (11)$$

Remark. The upper-bounded TPN derived in (11) may not be the exact value of the TPN, because if the corresponding extended graph of p_{min} in (11) is not the next constructed graph G_{i+1}, then the true TPN should be smaller (by the decreasing property of $Y_i(G_e)$ in the proof). That is why we name τ_i^* the upper-bounded TPN.

4.1.4 Calculation of the Accumulated State-Change Probability

According to (8), the accumulated state-change probability is given by

$$X_{i-1} = \prod_{j=1}^{i-1} P_{\tau_i^*}(C_j \to C_{j+1}) = \begin{cases} X_{i-2} \times P_{\tau_{i-1}^*}(C_{i-1} \to C_i), & i > 1, \\ 1, & i = 1. \end{cases}$$

Since the state-change probability is not derived, we opt to calculate the accumulated state-change probability after the state of the rectified graph reconstruction procedure has been changed.

Let us consider the scenario that the constructed graph is changed from G_{i-1} to G_i. After the state has been changed, the probability $P(G_i = G_e)$ becomes either one or zero for every extended graph G_e, and this means that

$$P(G_i = G_e) = \begin{cases} 0, & G_e \in \mathcal{G}(G_{i-1}) - \{G_i\}, \\ 1, & G_e = G_i. \end{cases} \quad (12)$$

Then, the state-change probability $P_{\tau_{i-1}^*}(C_{i-1} \to C_i)$ becomes

$$\begin{aligned} & P_{\tau_{i-1}^*}(C_{i-1} \to C_i) = \\ & \sum_{G_e \in \mathcal{G}(G_{i-1})} P_{\tau_{i-1}^*}(C_{i-1} \to C_i \mid G_i = G_e) \times P(G_e = G_i) \\ &= P_{\tau_{i-1}^*}(C_{i-1} \to C_i \mid G_i = G_i) \times P(G_i = G_i) \qquad \text{(by (12))} \\ &= 1 - \Big(1 - P(T(G_i) = e_i)\Big)^{\tau_{i-1}^*}, \qquad \text{(by (9))} \end{aligned}$$

where e_i is the new edge added to G_i.

Hence, the accumulated state-change probability X_{i-1} can be obtained after the rectified graph reconstruction procedure has proceeded from states C_{i-1} to C_i. The calculation of the accumulated state-change probability is presented as follows:

$$X_{i-1} = \begin{cases} X_{i-2} \times \left(1 - \Big(1 - P(T(G_i) = e_i)\Big)^{\tau_{i-1}^*}\right), & i > 1, \\ 1, & i = 1. \end{cases} \quad (13)$$

4.1.5 The Accumulated State-Change Probability for a Disconnected State

We now consider the case when the assumption that the constructed graph is always connected is removed, that is, a normal execution of the RPPM algorithm. Supposing that the rectified graph reconstruction procedure enters the disconnected state D_{i+1} from the connected state C_i, the update of the accumulated state-change probability has to be changed.

According to the previous discussion, the accumulated state-change probability depends on the constructed graph in state D_{i+1}, which is disconnected. Nevertheless, because

TPN_subroutine(Graph G, Traceback Confidence Level P^*)

/* Let the variables τ, X and p_min be static variables, which mean the values of these variables are not erased after exiting the subroutine. */

1. **If** G is not connected & $G.edge > 0$; **then**
2. **If** the previous state is a connected state ; **then**
3. $X := X \times (1 - (1 - p\_min)^{\tau})$;
4. **end If**
5. exit the subroutine ;
6. **end If**
7. **If** the previous state is a connected state & $G.edge > 0$; **then**
8. p := packet-type probability of the new edge of the constructed graph ;
9. $X := X \times (1 - (1 - p)^{\tau})$;
10. **end If**
11. $p\_min := 1$;
12. **Foreach** extended graph G_e in $\mathcal{G}(G)$; **do**
13. p := the packet-type probability of the extended edge of G_e ;
14. $p\_min := min(p\_min, p)$;
15. **end Foreach**
16. $\tau := \lfloor \log(1 - P^*/X)/\log(1 - p\_min) + 1 \rfloor$;
17. **return** τ;

Fig. 11. The pseudocode of the TPN calculation subroutine.

the graph G_i is disconnected, the packet-type probability $P(T(G_i) = e_i)$ cannot be found. As an alternative, we choose $\min_{G_e \in \mathcal{G}(G_i)} P(T(G_e) = e')$ in (11) as the value of $P(T(G_{i+1}) = e_{i+1})$ in (13). The reason for the above choice is given as follows:

$$\tau_i^* > \frac{\log\left(1 - \frac{P^*}{X_{i-1}}\right)}{\log\left(1 - p_{min}\right)} \Rightarrow X_{i-1} \times \left(1 - \left(1 - p_{min}\right)^{\tau_i^*}\right) > P^*,$$

where $p_{min} = \min_{G_e \in \mathcal{G}(G_i)} P(T(G_e) = e')$.

Hence, the accumulated state-change probability is still larger than the traceback confidence level P^* by choosing $\min_{G_e \in \mathcal{G}(G_i)} P(T(G_e) = e')$ as the value of $P(T(G_{i+1}) = e_{i+1})$ in (13). In the next section, we conclude this section and provide the pseudocode of the TPN calculation subroutine.

4.2 Section Summary and Termination Packet Number Calculation Subroutine

To summarize, we have presented how one can calculate the TPN at every connected state of the graph construction procedure so that the RPPM algorithm returns a correct constructed graph with a specified probability P^*.

Fig. 11 shows the subroutine that calculates the TPN, and it is executed whenever the rectified graph reconstruction procedure enters a new state. When the routine is visited for the first time, the variable "X" that is used to store the accumulated state-change probability is initialized to one. Next, based on the connectivity of the current constructed graph, the variable "X" is updated in different ways: 1) if the current constructed graph is connected, the subroutine calculates the packet-type probability of the new edge and then updates the variable "X," and 2) if the current constructed graph is disconnected, the subroutine uses the minimum packet-type probability of the extended edge that was chosen from the extended graphs of the previous constructed graph, that is, "p_min" in the pseudocode in Fig. 11. Next, if the current constructed graph is disconnected, the TPN subroutine will not calculate the TPN, and one should exit the subroutine. Otherwise, the subroutine calculates the TPN based on (11). Finally, the subroutine returns the calculated TPN.

5 Simulation Results

In this section, we present the simulation results to show that the RPPM algorithm is able to guarantee the correctness of the constructed graph, independent of the marking probability and the structure of the attack graph. First, we describe the simulation environment.

5.1 The Simulation Environment

Every simulation of the RPPM algorithm starts with a testing network rooted at the victim, that is, the attack graph. The configuration of the network follows the assumption stated in Section 2.1. In addition, the network has at least one leaf router, that is, a router with zero incoming edges. Each edge between two routers is directed and is assumed to have infinite capacity. Thus, no packet is lost under this environment.

Next, we describe the properties of the simulated packets. All packets are homogeneous in terms of type, size, etc. Every packet's destination is set to the victim, and every packet starts its itinerary at one of the leaf routers of the testing network chosen at random. Further, the paths traversed by the packets are chosen at random.

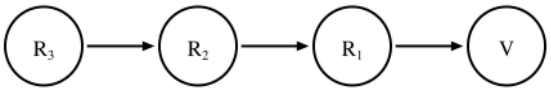

Fig. 12. An example linear network with three edges.

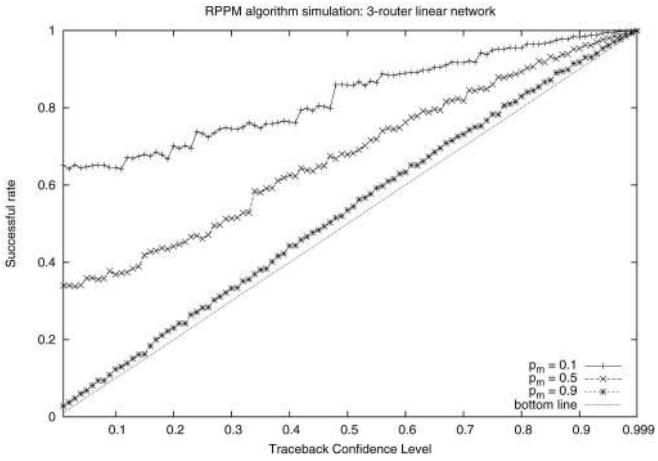

Fig. 13. The simulations show that the larger the marking probability is, the closer to the worst-case execution the simulation result becomes.

5.2 Simulation: Different Values of the Marking Probability

In this set of simulations, the impact of the marking probability on the successful rate of the RPPM algorithm will be studied. As presented in Section 3, the marking probability is one of the factors that determines the packet-type probability and also the termination packet number. As a matter of fact, the marking probability is closely related to the occurrences of the different execution scenarios described in Section 2.3.3.

A high value of the marking probability is analogous to the worst-case scenario. If the value of the marking probability is high, most of the arrived packets are encoding edges that are close to the victim. Then, the constructed graph is always connected with a very high probability, and thus, this case is analogous to the worst-case scenario. On the contrary, the execution of the RPPM algorithm is close to the best-case scenario with a very low value of the marking probability.

We have conducted a set of simulations to verify the above claims. In this set of simulations, the testing network is the network depicted in Fig. 12. The simulations are performed at three different values of the marking probability: 0.1, 0.5, and 0.9. The RPPM algorithm is repeated 10,000 times in order to generate one data point, and each data point is obtained by dividing the number of successful executions by the total number of executions of the RPPM algorithm.

The results of the simulations are shown in Fig. 13. In the figure, in spite of the simulation results, there is an extra plot in the figure named the "bottom line," which represents the function $y = x$. Since we expect that the successful rate should be larger than the traceback confidence level, no data point should appear below the bottom line. We now analyze the simulation result. First, all the data points are above the bottom line, and this shows that the RPPM algorithm can guarantee the correctness of the constructed graph under different values of the marking probability. Second, one can observe that as the marking probability increases, the rate at which the RPPM algorithm returns a correct graph decreases.

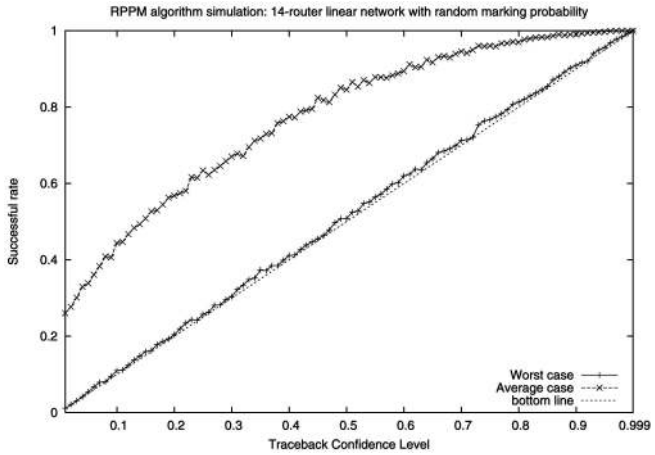

Fig. 14. RPPM algorithm simulation: 14-router linear network with random marking probability.

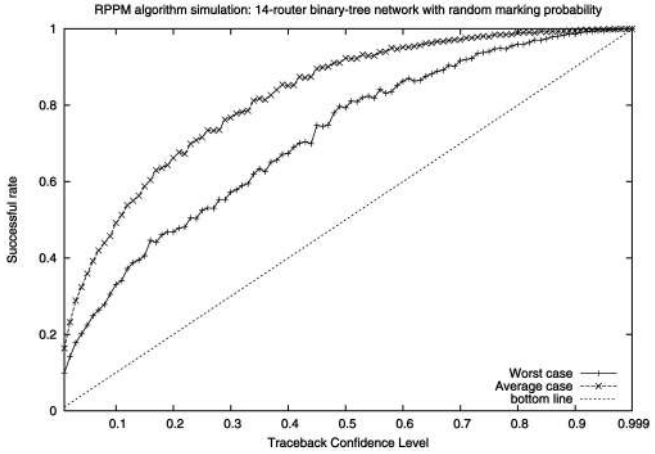

Fig. 15. RPPM algorithm simulation: 14-router binary-tree network with random marking probability.

With $p_m = 0.9$, the plot is very close to the bottom line, which implies the worst-case scenario. Through this set of simulations, we showed that the RPPM algorithm can guarantee the correctness of the constructed graph under different values of the marking probability.

5.3 Simulation: Different Graph Structures

The second set of simulations tests if the RPPM algorithm can guarantee the promised successful rate under different graph structures. In this set of simulations, we execute the simulations under both the worst-case and the average-case scenarios. The worst-case scenario is forced to be happening by restricting the packet generation process, whereas the average-case scenario is a normal execution of the RPPM algorithm without any constraints. In addition, for each execution of the RPPM algorithm, the marking probability is inclusively set to a random number from 0.1 to 0.9.

The simulation results for the linear network, the binary-tree network, and the random-tree network that contain 14 routers and one victim are shown in Figs. 14, 15, and 16, respectively. The topologies of the linear and the binary-tree networks are self explanatory, and a random-tree network means that the nodes are randomly connected with the following constraints:

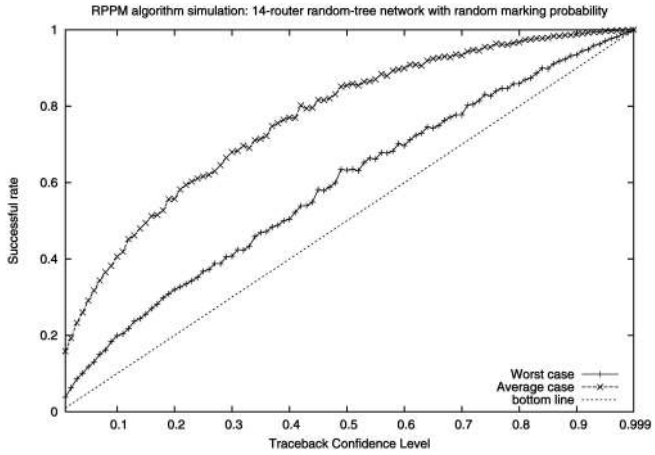

Fig. 16. RPPM algorithm simulation: 14-router random-tree network with random marking probability.

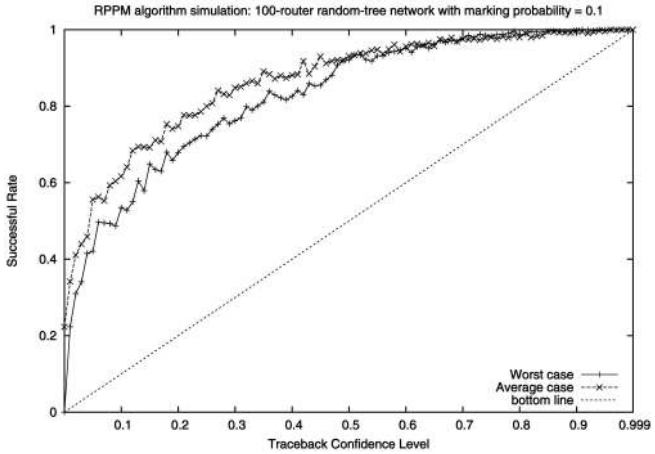

Fig. 17. RPPM algorithm simulation: 100-router random-tree network, with $\text{marking probability} = 0.1$.

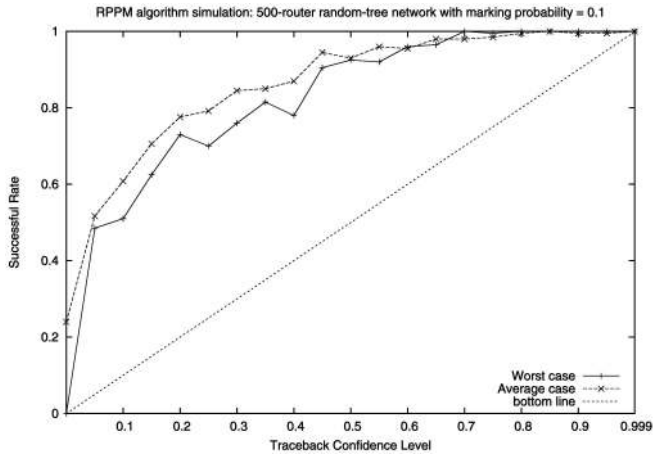

Fig. 18. RPPM algorithm simulation: 500-router random-tree network, with $\text{marking probability} = 0.1$.

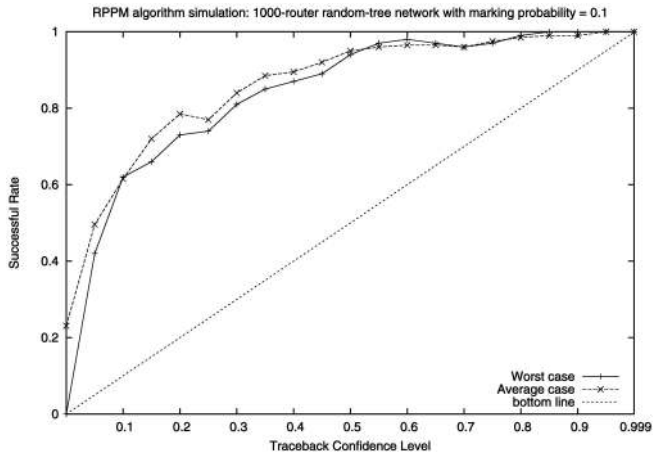

Fig. 19. RPPM algorithm simulation: 1,000-router random-tree network, with $\text{marking probability} = 0.1$.

1. Every router can reach the victim in a nonzero number of hops.
2. There must be no cycles in the graph.
3. The victim must not have any outgoing edges.
4. Every router can only have one outgoing edge.

In addition, as Paxson [18] suggested, the longest router in the Internet is 32. Then, the maximum length of the paths of the testing network is therefore 32.

All three results show that no matter what the network is, all the data points are above the bottom line. Hence, this shows that the RPPM algorithm guarantees the correctness of the constructed graph, independent of the structure of the real network graph. In addition, the simulation results support the claim that the average-case scenario outperforms the worst-case scenario in terms of the successful rate. Furthermore, we extend the simulations on the random-tree network to larger network scales with 100, 500, and 1,000 routers, and the results are shown in Figs. 17, 18, and 19, respectively. According to the results, the increasing network scale does not affect the guarantee provided by the RPPM algorithm.

In conclusion, the simulation results showed that the RPPM algorithm guarantees the correctness of the constructed graph, independent of the marking probability and the structure of the attack graph.

6 Supporting Routers with Multiple Victim Routes

In this section, we relax the assumption that every router has only one outgoing route toward the victim. This change may cause the attack packets to take more than one path toward to the victim, and the routers in the constructed graph may have more than one outgoing edge.

In the following, we first discuss the problem that emerged when the RPPM algorithm is applied to routers that have multiple victim routes. In addition, a set of simulations is performed to illustrate the severity of the problem. Second, we present the solution to the problem caused by the relaxed assumption: the method introduces an extra set of extended graphs. Last, we perform simulations based on this solution and compare the results with and without the support of multiple victim routes.

6.1 Problem of Multiple Victim Routes

Originally, without considering routers that have multiple victim routes, the arrival of a new encoded edge will add only a new node and a new edge to the constructed graph (note that it is the worst-case execution scenario). However, when we allow a router to have multiple victim routes, the arrival of a marked packet that encodes a new edge can result in two different scenarios: 1) a new node is added,

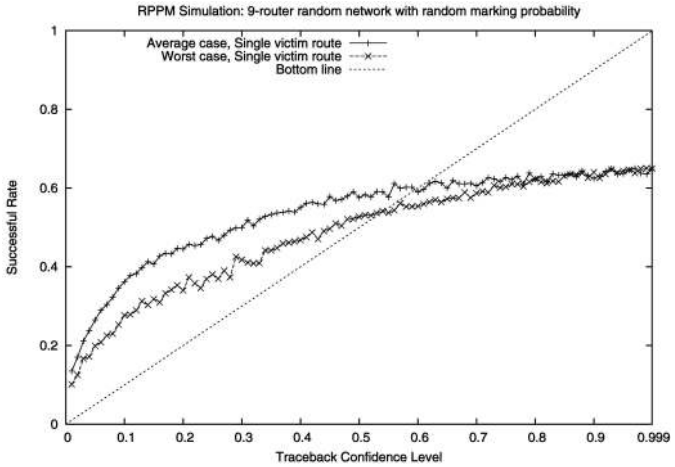

Fig. 20. When the routers have more than one victim route, the RPPM algorithm cannot guarantee the correctness of the constructed graph when the confidence level is larger than 0.59.

that is, one node plus one edge and 2) no new node is added, which means that the new edge connects two existing nodes. Since the latter case is not considered by the RPPM algorithm, one may then doubt the guarantee of the successful rate of the RPPM algorithm. The following simulation supports this doubt.

6.1.1 The Simulation Environment

The testing network is a random-tree network with 10 nodes: one victim plus nine routers. However, this time, we allow the routers in the testing network to have more than one victim route. Again, the marking probability is set to a random number in [0.1: 0.9], and the values are the same for all routers.

6.1.2 The Simulation Result

Fig. 20 shows the simulation results for both the average-case and the worst-case executions. For small values of the traceback confidence level, the successful rates of both execution modes are still over the bottom line. However, the successful rate of the worst-case execution falls below the bottom line when the traceback confidence level goes beyond 0.54, whereas the successful rate of the average-case execution falls below the bottom line when the traceback confidence level goes beyond 0.59.

One can conclude that the RPPM algorithm cannot provide a guarantee of the successful rate in reconstructing the attack graph when the routers have multiple outgoing routes toward the victim.

6.2 Formulating an Extra Set of Extended Graphs

To solve the problem, we suggest introducing an extra set of extended graphs. The new set of extended graphs is defined as follows:

Definition 3. *Let $\mathcal{G}'(G_i)$ be the set of extended graphs of the constructed graph $G_i = (V_i, E_i)$ that supports multiple outgoing routes toward the victim:*

$$\mathcal{G}'(G_i) = \{G'_e = (V_i, E'_e) \mid \exists (u,v) \notin E_i \ \& \ u, v \in V_i \text{ such that } E'_e = E_i \cup (u,v)\},$$

and all graphs in $\mathcal{G}'(G_i)$ must not have any cycles.

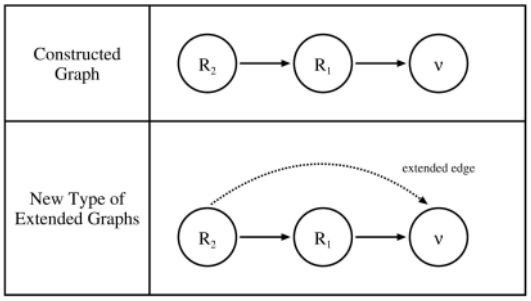

Fig. 21. An illustration of the extended graph with the support of multiple victim routes.

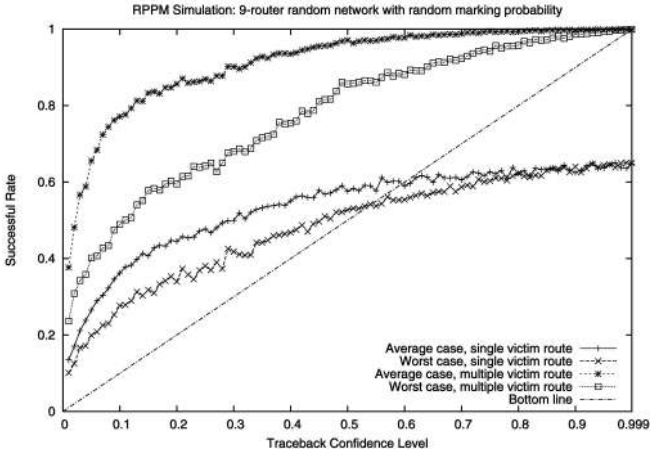

Fig. 22. With the support for multiple victim routes, the RPPM algorithm can provide the guarantee of the correctness of the constructed graph.

According to Definition 3, an extended graph in $\mathcal{G}'(G_i)$ introduces an extra edge to the constructed graph without an extra node. The edge connects any two existing nodes with two restrictions: 1) no cycles and 2) a multigraph should not be formed. Then, this definition creates a family of extended graphs with routers that have multiple victim routes.

We illustrate the definition of the new set of extended graphs through an example in Fig. 21. The upper part of the figure shows a constructed graph with two routers R_1 and R_2 and the victim v, and the lower part of the figure is the new extended graph. For this example, there can only be one extra edge (R_2, v) according to Definition 3.

6.3 Simulation: Support for Multiple Victim Routes

Definitions 2 and 3 together form an expanded set of extended graphs. We conduct the previous simulation again by using the expanded set of extended graphs, and the results are shown in Fig. 22. In this figure, the RPPM algorithm can guarantee the correctness of the constructed graph, again, with the support of multiple victim routes. Technically speaking, the introduction of the extra set of extended graphs actually increases the value of the TPN. As the TPN increases, the successful rate therefore increases.

6.4 Section Summary

In conclusion, we provided support for routers that have multiple victim routes. Such support is done through an expansion of the set of the extended graphs. We performed simulations to contrast the performances of the RPPM algorithm with and without such support.

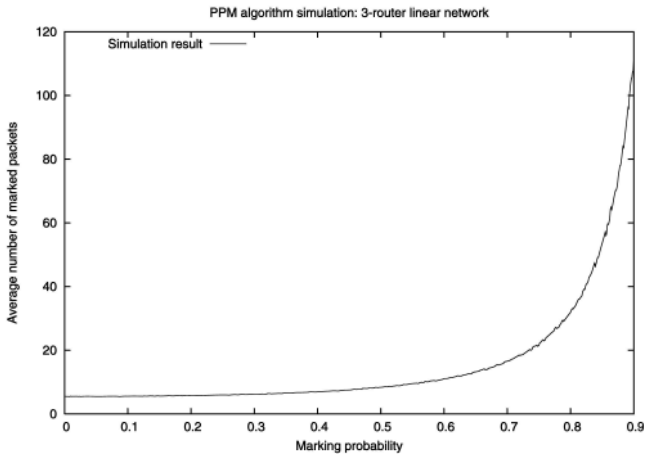

Fig. 23. The plot of the average number of marked packets required for a correct graph reconstruction against different values of the marking probability.

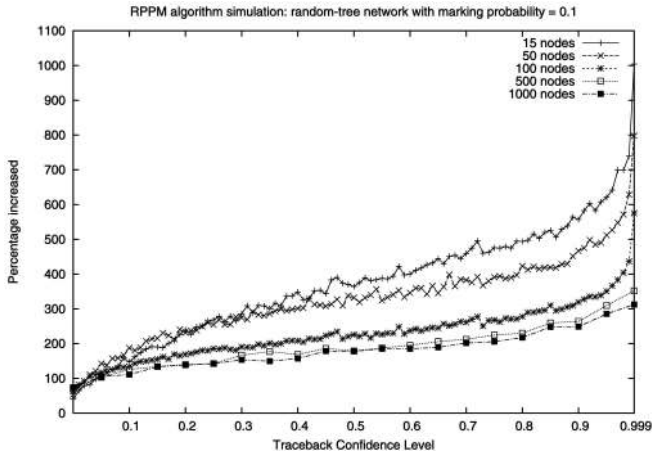

Fig. 24. The percentage of number of packets increases when the RPPM algorithm is compared to the PPM algorithm with different network scales.

The drawback of this support is computation. Let n be the number of nodes and m be the number of edges of the constructed graph. Originally, the number of extended graphs is of order $O(n)$. With the mentioned support, the order of the number of extended graphs becomes $O(nm)$. Hence, more time is spent on calculating the TPN at each connected state of the rectified graph reconstruction procedure. This shows the trade-off in handling routers with multiple victim routes.

7 Deployment Issues of the Rectified Probabilistic Packet Marking Algorithm

In this section, we discuss several issues in deploying the RPPM algorithm. We first discuss the choice in the marking probability. Then, we cover the trade-off of the RPPM algorithm over the PPM algorithm. Last, we address the scalability problem in the PPM and the RPPM algorithms.

7.1 Choice of the Marking Probability

It is not desirable to have a high value of the marking probability. First, a high value of the marking probability means a low value for the packet-type probabilities for the majority of the types of packets. Hence, this implies that a large number of marked packets are needed before the RPPM algorithm stops. This also implies a long execution time of the RPPM algorithm.

Let us take a linear network with three routers and one victim (as shown in Fig. 12) as an example to illustrate the relationship between the marking probability and the number of packets required. Fig. 23 shows the result of a simulation that aims at counting the average number of marked packets required for a correct graph reconstruction with different values of the marking probability. The result shows that for small values of marking probability, the number of required packets is small. Nevertheless, the number of required packets dramatically increases for large values of the marking probability.

Despite the above reason, according to Section 5, a high value of the marking probability implies the presence of the worst-case scenario of the RPPM algorithm. Although the worst-case scenario can still guarantee the successful rate, it would be more beneficial to set the value of the marking probability to a lower value so as to gain a larger successful rate than what is expected.

In conclusion, one should choose a small value for the marking probability for a faster and more reliable graph reconstruction. Note that there would be a large number of unmarked packets if one chooses a too-small value of the marking probability.

7.2 Execution Time Comparison between the PPM and the RPPM Algorithms

In order to guarantee the correctness of the constructed graph, the RPPM algorithm has to collect extra packets so as to attain such a guarantee. Technically speaking, before the moment that the constructed graph becomes the same as the attack graph, the number of marked packets collected should be the same for both the PPM and RPPM algorithms. After the constructed graph has become the attack graph, the RPPM algorithm has to wait until the number of collected packets is larger than the TPN. In other words, that extra sum of packets is the trade-off in deploying the RPPM algorithm than the PPM algorithm.

However, it is difficult to determine a theoretical value or bound of the TPN, because the TPN calculation depends on the construction process of the constructed graph. The construction process, in turn, depends on the sequence of the arrivals of the marked packets, which is randomized. Alternatively, we conduct an empirical study on the trade-off of the RPPM algorithm.

In Fig. 24, we present the number of increased marked packets when one compares the number of packets collected by the RPPM algorithm to those collected by the PPM algorithm (which is instructed to stop when the constructed graph becomes the attack graph). Such a set of simulations is performed using a marking probability of 0.1 (as suggested in Section 7.1) with increasing network scales: from a 15-node random-tree network to a 1,000-node one. The RPPM algorithm is operated under the average-case scenario.

Three main observations can be concluded from this set of simulations. First, when the traceback confidence level increases, the trade-off of the RPPM algorithm increases. Second, the number of collected packets by the RPPM algorithm is larger than those collected by the PPM algorithm by several times for the small range of the traceback confidence level (two to five times for the traceback

TABLE 1
The Average Number of Packets and the Time Required to Reconstruct a Correct Constructed Graph in a 100BaseT Ethernet

Number of Nodes	15	50	100	500	1000
Average number of marked packets required for PPM algorithm	92	1170	1550	15222	40792
Average time required in 100BaseT Ethernet (in second)	0.011	0.140	0.180	1.820	4.910

confidence level below 0.8), and such an increase reaches 10 times for high values of the traceback confidence level.

Last, an interesting observation is that the trade-offs for small networks are more significant than those for large networks. This can be explained by the probability of forming a disconnected graph. For a large network, such a probability is much higher than that of a small network. When a disconnected graph is formed, the TPN calculation is skipped until the graph becomes connected. Hence, this keeps the value of the TPN small during the ending states of the RPPM algorithm.

On the other hand, according to Table 1, one can observe that the time for the PPM algorithm to collect enough packets is in the order of a few seconds in a 100BaseT Ethernet.[1] Therefore, although the trade-off of the RPPM algorithm could reach a multiple of 10, such a trade-off is acceptable.

7.3 Scalability

Scalability is one of the weaknesses of the PPM algorithm. One can observe that as the path length between the victim and the leaf router becomes longer, it becomes more difficult to collect a complete set of the marked packets. The case is that not only the path length affects the traceback time but the size of the attack graph also matters. In Fig. 25, one can observe that the number of marked packets required to build the constructed graph increases with the size of the graph, and the trend does not subside. Therefore, the PPM algorithm itself has a scalability problem. Nonetheless, as the RPPM algorithm inherits the packet marking procedure from the PPM algorithm, the RPPM algorithm also has the scalability problem.

As suggested in Section 7.2, for small networks, the traceback process takes only a few seconds to complete. However, for networks as large as the one in [19] (with nearly 200,000 routers and more than 600,000 directed links), the traceback process may take days to finish.

8 Conclusion and Future Work

In this work, we have pinpointed that the PPM algorithm lacks a proper definition of the termination condition. Meanwhile, using the expected number of required marked packets $E[X]$ as the termination condition is not sufficient. The above two outstanding problems only lead to an undesirable outcome: there is no guarantee of the correctness of the constructed graph produced by the PPM algorithm.

We have devised the *rectified graph reconstruction procedure* to solve the above two problems, and we name the new traceback approach the *RPPM* algorithm. The RPPM algorithm, on one hand, does not require any previous knowledge about the network graph. On the other hand, it guarantees that the constructed graph is a correct one, with a specified probability, and such a probability is an input parameter of the algorithm.

1. Under a 100BaseT Ethernet, one can transmit at most 8,333 packets (each with 1,500 bytes) in 1 s.

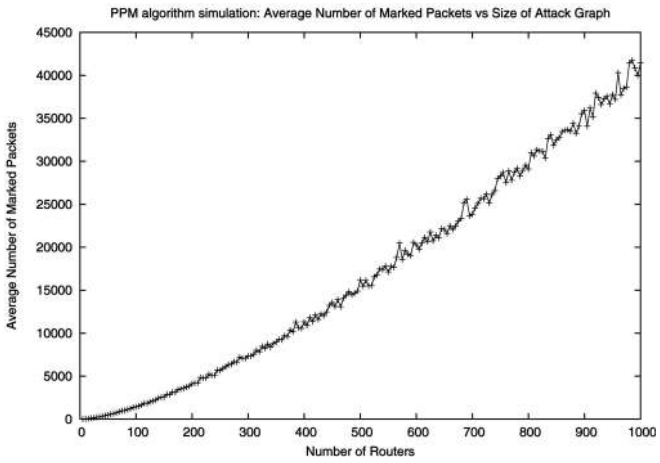

Fig. 25. Scalability analysis: average number of marked packets collected by the PPM algorithm versus the size of the attack graph.

We have carried out a series of simulations to show the correctness and the robustness of the RPPM algorithm. The simulation results show that the RPPM algorithm can always satisfy our claim that the constructed graph is correct with a given probability. In addition, the algorithm is robust under different values of the marking probability and different structures of the attack graphs. To conclude, the RPPM algorithm is an effective means of improving the reliability of the original PPM algorithm.

Since the RPPM algorithm is an extension of the PPM algorithm, the RPPM algorithm inherits defects of the PPM algorithm. Problems such as scalability and different attack patterns will be future research directions.

Acknowledgments

The authors would like to thank the editor and supporting staff for coordinating the review process. They also thank the anonymous reviewers for their insightful comments and constructive suggestions. The work of M.H. Wong was partially supported by the RGC Grant 4208/04E. The work of John C.S. Lui was supported in part by the RGC Grant 2150347.

References

[1] "CERT Advisory CA-2000-01: Denial-of-Service Developments," Computer Emergency Response Team, http://www.cert.org/-advisories/-CA-2000-01.html, 2006.
[2] J. Ioannidis and S.M. Bellovin, "Implementing Pushback: Router-Based Defense against DDoS Attacks," *Proc. Network and Distributed System Security Symp.*, pp. 100-108, Feb. 2002.
[3] S. Bellovin, M. Leech, and T. Taylor, *ICMP Traceback Messages,* Internet Draft Draft-Bellovin-Itrace-04.txt, Feb. 2003.
[4] K. Park and H. Lee, "On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets," *Proc. ACM SIGCOMM '01*, pp. 15-26, 2001.
[5] P. Ferguson and D. Senie, "RFC 2267: Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing," *The Internet Soc.*, Jan. 1998.

[6] D.K.Y. Yau, J.C.S. Lui, F. Liang, and Y. Yam, "Defending against Distributed Denial-of-Service Attacks with Max-Min Fair Server-Centric Router Throttles," *IEEE/ACM Trans. Networking,* no. 1, pp. 29-42, 2005.
[7] C.W. Tan, D.M. Chiu, J.C. Lui, and D.K.Y. Yau, "A Distributed Throttling Approach for Handling High-Bandwidth Aggregates," *IEEE Trans. Parallel and Distributed Systems,* vol. 18, no. 7, pp. 983-995, July 2007.
[8] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical Network Support for IP Traceback," *Proc. ACM SIGCOMM,* pp. 295-306, 2000.
[9] D. Dean, M. Franklin, and A. Stubblefield, "An Algebraic Approach to IP Traceback," *ACM Trans. Information and System Security,* vol. 5, no. 2, pp. 119-137, 2002.
[10] D.X. Song and A. Perrig, "Advanced and Authenticated Marking Schemes for IP Traceback," *Proc. IEEE INFOCOM '01,* pp. 878-886, Apr. 2001.
[11] A.C. Snoeren, C. Partridge, L.A. Sanchez, C.E. Jones, F. Tchakountio, S.T. Kent, and W.T. Strayer, "Hash-Based IP Traceback," *Proc. ACM SIGCOMM '01,* pp. 3-14, Aug. 2001.
[12] K. Park and H. Lee, "On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial-of-Service Attacks," *Proc. IEEE INFOCOM '01,* pp. 338-347, 2001.
[13] K.T. Law, J.C.S. Lui, and D.K.Y. Yau, "You Can Run, But You Can't Hide: An Effective Methodology to Traceback DDoS Attackers," *IEEE Trans. Parallel and Distributed Systems,* vol. 15, no. 9, pp. 799-813, Sept. 2005.
[14] M. Adler, "Trade-Offs in Probabilistic Packet Marking for IP Traceback," *J. ACM,* vol. 52, pp. 217-244, Mar. 2005.
[15] H. von Schelling, "Coupon Collecting for Unequal Probabilities," *Am. Math. Monthly,* vol. 61, pp. 306-311, 1954.
[16] C. Hedrick, "RFC 1058: Routing Information Protocol," *The Internet Soc.,* June 1988.
[17] J. Moy, "RFC 2328: Open Shortest Path First (OSPF) Version 2," *The Internet Soc.,* Apr. 1998.
[18] V. Paxson, "End-to-End Routing Behavior in the Internet," *IEEE/ACM Trans. Networking,* vol. 5, pp. 601-615, Oct. 1997.
[19] "CAIDA Router-Level Topology Measurements," Cooperative Assoc. Internet Data Analysis, http://-www.caida.org/-tools/measurement/skitter/router_topology/, 2006.

Tsz-Yeung Wong received the PhD, MPhil, and BSc degrees all from the Department of Computer Science and Engineering at the Chinese University of Hong Kong in 2007, 2002, and 2000, respectively. He joined the Chinese University of Hong Kong in August 2007 as an instructor. His research interests include distributed algorithms, networking, and computer and network security.

Man-Hon Wong received the BSc and MPhil degrees from the Chinese University of Hong Kong in 1987 and 1989, respectively, and the PhD degree from the University of California at Santa Barbara in 1993. He joined the Chinese University of Hong Kong in August 1993 as an assistant professor and was promoted as an associate professor in 1998. His research interests include transaction management, mobile databases, data replication, distributed systems, and computer and network security.

Chi-Shing (John) Lui received the PhD degree in computer science from the University of California, Los Angeles (UCLA). After his graduation, he joined the IBM Almaden Research Laboratory/San Jose Laboratory and participated in various R&D projects on file systems and parallel I/O architectures. He later joined the Department of Computer Science and Engineering, Chinese University of Hong Kong (CUHK). He is an associate editor for the *Performance Evaluation Journal*, the *IEEE Transactions on Computers*, and the *IEEE Transactions of Parallel and Distributed Systems*. He was a TPC cochair of ACM Sigmetrics 2005 and a general cochair of the 15th IEEE International Conference on Network Protocols (ICNP 2007). His research interests include system and in theory/mathematics, in particular theoretic/applied topics in data networks, distributed multimedia systems, network security, OS design issues, and mathematical optimization and performance evaluation theory. His personal interests include films and general reading. He is a member of the ACM, a senior member of the IEEE, an elected member of the IFIP WG 7.3, and the vice president of ACM Sigmetrics. He received various departmental teaching awards and the CUHK Vice Chancellor's Exemplary Teaching Award. He is a corecipient of the Best Student Paper Award in the 24th IFIP WG 7.3 International Symposium on Computer Performance, Modeling, Measurements and Evaluation (Performance 2005) and the IEEE/IFIP Network Operations and Management Symposium (NOMS).