Journal of Information Processing Vol.23 No.5 613–622 (Sep. 2015)

[DOI: 10.2197/ipsjjip.23.613]

Regular Paper

METRO: Measurement of End-to-End Route Trust

Nasato Goto[1,a)] Akira Kanaoka[2,b)] Masayuki Okada[3,c)] Eiji Okamoto[1,d)]

Received: November 27, 2014, Accepted: March 4, 2015

Abstract: Given the current security situation on the Internet, it is important to determine the trust of the communication routes between a client and server. However, such determination can only be established by end terminals such as clients and servers, not by intermediate routers or network providers so far. Revelations regarding PRISM and other programs highlight the importance of this issue. In this paper, a method to identify the trust level of a route between a client and a server is proposed that uses packet authentication, Probabilistic Packet Marking (PPM), and knowledge bases maintained by trusted third parties. A prototype system of the proposed method was developed and evaluated, prove its feasibility. To the best of our knowledge, the proposed method is the first for identifying the trust level of a route based on information obtained from intermediate routers or Autonomous Systems (ASs).

Keywords: network security, route trust, probabilistic packet marking

1. Introduction

PC and smartphone applications are increasingly utilizing data communication services, increasing Internet usage worldwide. Problems related to information security when using the Internet are also increasing, and solutions for such problems are urgently needed.

Current end-to-end communication on the Internet when using a service can cross many regions and countries; however, it is not known which regions and countries are being crossed during a particular session. This issue is not limited to regions and countries; it is also unknown which networks or routers are being used during a session.

In 2013, PRISM, the surveillance program of the US National Security Agency (NSA), became widely known because of disclosures made by the Guardian and the Washington Post newspapers [1], [2]. Although the implicated providers and companies denied joining the program, in the news it was reported otherwise. Furthermore, other similar incidents have occurred. In December 2013, the NSA asked that RSA's pseudo-random number generator have a backdoor by default [3]. Additionally, in May 2014, it was revealed that the NSA put a backdoor for surveillance on routers and servers exported overseas [4].

The use of encryption and mutual authentication for end-to-end communication on the Internet is currently widespread. Secure Socket Layer/Transport Layer Security (SSL/TLS) on HTTP is the most common technology for encryption and mutual authentication. Given the current situation, the determination of the communication routes between clients and servers is also desirable. In addition, this determination can be established only by end terminals, such as clients and servers, not by intermediate routers or network providers.

For determining the route between a client and a server, traceroute is a widely known method. Its main drawbacks are its computational load and inability to obtain return journey information. Reverse traceroute obtains return journey information, but its performance remains inefficient. Secure traceroute was proposed for avoiding man-in-the-middle attacks by intermediate routers. This method uses cryptographic techniques to identify packets between routers and hide data from third parties. Although not currently known, its performance may be a drawback. To the best of our knowledge, no method that identifies route trust level based on information obtained from intermediate routers or Autonomous Systems (ASs) exists.

In this paper, a method to determine the trust level of the route between a client and a server is proposed. This method includes packet authentication between routers and obtains route information between a client and server using Probabilistic Packet Marking (PPM), an IP traceback technique.

In our study, a prototype of the proposed method was developed using a modified Linux kernel and specific communication between client and server. The performance of the proposed method was then evaluated. The results show that the method does not heavily load the routers.

The remainder of this paper is organized as follows. In Section 2, we describe the outline of our proposed method and its requirements. A route detection method is presented in Section 3, as well as related systems, such as PPM and traceroutes. We define trust level and introduce a barometer to calculate it in Section 4. In Section 5, we discuss the security of the proposed method. The implementation and performance evaluation of the proposed method are described in Section 6. In Section 7, the limitations of the proposed method and future work are presented. The paper is concluded in Section 8.

[1] University of Tsukuba, Tsukuba, Ibaraki 305–8573, Japan
[2] Toho University, Funabashi, Chiba 274–8510, Japan
[3] Japan Network Information Center, Chiyoda, Tokyo 101–0047, Japan
[a)] goto@cipher.risk.tsukuba.ac.jp
[b)] akira.kanaoka@is.sci.toho-u.ac.jp
[c)] okadams@nic.ad.jp
[d)] okamoto@risk.tsukuba.ac.jp

© 2015 Information Processing Society of Japan

2. Proposed Method

In this section, we introduce our proposed method and its requirements. It consists of two elements: route detection and trust level estimation. Details of each element are given in Sections 3 and 4.

2.1 Outline of Proposed Method

Figure 1 shows the outline of the proposed method and indicates the flow that a client uses to identify the trust level of a route to a server. The black routers in the figure are those that have implemented the proposed method, and the gray ones those that have not.

(1) Each black supported router marks the data, which have been divided and stored in the routers in advance, on the packets sent from the server.
(2) The client gathers the marked packets, checks the distance data, as described in the next section, and detects the number and approximate location of the gray routers on the route.
(3) The client reconstructs the router information from the marked data. Then it obtains IP and AS route information.
(4) The client queries the knowledge base, which includes trust information and cooperates with Internet router topology or country data, about the trust level using reconstructed reliability information.
(5) The knowledge base replies to the query.
(6) From the replies, the client estimates candidates for the gray routers.
(7) By synthesizing the total information that the client has obtained thus far the trust level of the route is determined.

We defined (1) to (3) as the route detection element and (4) to (7) as the trust level estimation element.

2.2 Requirements for Route Detection

A method that determines the trust of the route between end entities must meet several requirements. In this section, we list the important requirements for route detection.

2.2.1 Protection against Revealing Router Information to Other Entities

It is not desirable that intermediate router information be revealed to all entities, as occurs in the existing PPM scheme, since malicious users or routers can use this information for their own purposes. Therefore, information regarding intermediate routers must be provided only to legitimate and trusted users.

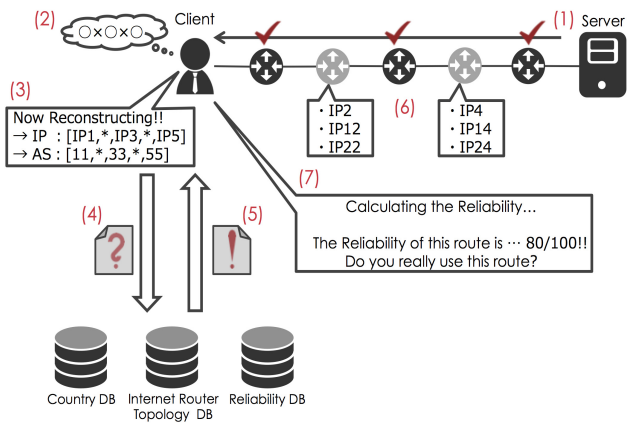


Fig. 1 Outline of proposed system.

2.2.2 Low Impact on Router Performance

We could add a function that is sufficiently rich to avoid all threats if infinite resources were available on Internet routers and networks. However, deployment of the method on network devices must have a low performance impact if the method is to be applied on intermediate routers.

2.2.3 Active Adversary Model

The semi-honest adversary or curious-but-honest adversary sees the communication data between a client and server but operates appropriately, and for this reason is often considered for implementing novel functions on network equipment. However, the underlying problem is that assuming a semi-honest adversary model is not acceptable. We must assume an active adversary model that can rewrite or overwrite any communication data between a client and server.

2.3 Requirements for Trust Level Estimation

To identify route trust, the trust level estimation element is essential. Some requirements are mentioned here, and the definition of trust level is given in Section 4.

2.3.1 Providing Trust Information from Trusted Third Party

In our method, the trust information has to be provided openly by a trusted third party. Moreover, the trust level should be not a static but rather a dynamic value that is updated according to social conditions, the revelation of serious problems, and the user's reputation.

2.3.2 Complementing the Proposed Method to Unsupported Nodes

Although in academic studies it is often assumed that all entities can readily adopt novel technologies, in practice, deploying a new method on all entities is extremely difficult; in addition, its deployment and migration status is not managed. Hence, the provision of trust information, even when only some entities deploy the novel technology, is required.

2.3.3 Reflecting User's Own Policy in Trust Level Estimation

The policy of the client is an important factor in refining the utilization of trust information. It is not necessary to use the entire trust level provided by the trusted third party, but rather to apply the given trust information only the user's intrinsic policy. This facilitates the user's decision whether to accept the reconstructed route.

3. Route Detection Method

We describe the *Route Detection* method in this section. First, we explain existing systems such as PPM and traceroute. And then, the prior experiment and system specifications of the proposed method are discussed.

3.1 Related Work

3.1.1 Probabilistic Packet Marking

Savage et al. proposed PPM in 2000 as an IP traceback

© 2015 Information Processing Society of Japan

method [5]. IP traceback is a countermeasure against a Denial of Service Attack (DoS Attack), the objective of which is to detect the source of the packets sent from the attacker to the victim. Various traceback techniques have been proposed to date, among which PPM is superior in many ways: it does not require a mechanism to monitor the network, generate unnecessary packets, or require extended information beyond the IP version 4 (IPv4) header. Thus, it has been actively discussed [6], [7], [8], [9], [10], [11], [12].

In the original PPM method proposed by Savage et al., the routers along the attack route probabilistically mark their own information onto packets so that victims can reconstruct the route from the attacker to themselves by collecting the marked packets.

Let us assume a router X on the route. X retains 64 bits of data that comprise of bit interleaving of 32-bit IP address and its 32-bit hash value. The 64-bit data is divided into eight fragments. When X marks the packets with static probability p, it marks 16 bits of information in the IPv4 *Identification* field. The information consists of three elements: a fragment (8 bits) randomly chosen from one of the eight fragments, the offset of that fragment (3 bits), and the distance metric from the marked router (5 bits).

The characteristics of this method include not only that single router information is marked, but also an exclusive OR (XOR) operation is performed with the next-hop router information. If the downstream router Y does not mark a packet, which occurs with probability $1 - p$, and if the distance metric on the packet is 0, then Y overwrites the information with the XOR of its own information and the information already written on the packet. Regardless of the distance metric value, Y also increments the distance on the packets. Savage et al. called this *Edge Sampling*. **Figure 2** shows the XOR process handled by routers on the attack route. The victim collects packets marked with A$\oplus$B, B$\oplus$C, C$\oplus$D, and D, with distance metric 0.

To reconstruct the routes, the victim gathers the marked packets and uses the distance information d. If the distance value d is 0, as for packets like D in Fig. 2, this indicates that the XOR operation has not been carried out. These packets determine the router closest to the victim. Hence, the victim can reconstitute D's IP information by combining the eight fragments. Furthermore, by computing the XOR of D's IP information and C$\oplus$D ($d = 1$), the victim can reconstitute C's IP information. By repeating this process, the victim can completely reconstruct the attack route.

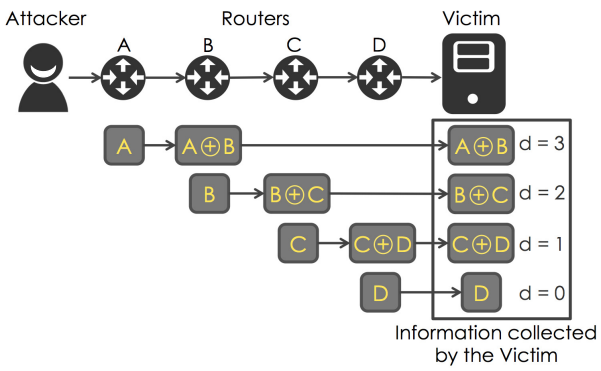


Fig. 2 Receiving a marked packet via original PPM.

After Savage et al.'s method was reported, Goodrich proposed another method called *randomize – and – link* in 2008 [6]. There are some differences between this method and the original: it does not divide the hash value, it uses a flexible data or hash size, and it does not include the distance value in the marking information. Hence, this method is a more general framework for PPM. In this method, the XOR operation is not performed. Hence, its objective is not to reconstruct attack routes, but to find quickly the routers next to the attacker. Moreover, it is claimed that the method can handle large amounts of data by dividing the marking process into two steps.

3.1.2 Traceroute

Traceroute is a well known tool for detecting packet routes [13]. In this method, the source host sends the destination host UDP or ICMP packets with the time to live (TTL) value incremented each time. Routers that receive these packets return an ICMP Time Exceeded Error packet to the source host. There is also a method called tcptraceroute that uses TCP [14].

Although the use of traceroute is widespread, there are many routers that do not reply to traceroute requests [17]. One reason is that generating ICMP reply packets could increase the load of the router and a second is that an organization may wish to conceal the IP address of routers for security reasons. Govindan and Paxson addressed the first reason by measuring router ICMP generation delays and found that there are fewer delays than expected [18]. However, they also reported that their measurements need scrutiny and improvement. Hence, we cannot ignore the possibility that some routers are unwilling to send ICMP Time Exceeded Errors because of the increased load.

To investigate this possibility, we measured the traceroute reply rate. For this measurement, we executed traceroute or tcptraceroute from a university laboratory (Univ. of Tsukuba in Japan) to Alexa's list of the top 1,000 domains [20]. We were able to reach only 386 domains. Furthermore, even when we reached the destination, we did not receive a response from 34.4% of the hops. On the other hand, when we used port 80 and tcptraceroute, 986 domains responded to our probe and 40.1% of the hops could not be seen.

Moreover, we can cite other defects of traceroute. First, traceroute cannot detect reverse route information. However, used on the Internet, the forward and reverse routes are very often different, and therefore, there is a large demand for determining the reverse routes (e.g., in network operation), which traceroute cannot meet. Second, Marchetta et al. pointed out the imprecision of traceroute's output [19]. Since sometimes the load balancers are overestimated and the routes changes misunderstood, traceroute can show users wrong routes in some cases.

To address the issue that traceroute can provide only the outbound route, Katz-Bassett et al. proposed a method called *reverse traceroute* in 2010 [16]. They spread several vantage points on the Internet that enabled users to detect the route from a destination host to a source host using the *Record Route* option in IPv4. In order to determine the route from destination host D to source host S, the vantage points send D ping packets, the *Record Route* option of which is active and the source IP disguised as S. Then, D sends a reply packet according to the spoofed source IP, and

© 2015 Information Processing Society of Japan

packets carry the *D* to *S* router information as a *Record Route*. Hence, this method reveals the inbound route. The authors implemented this method on the Internet. However, it required approximately 41 s (median) to complete the process, and hence, they reported that further improvement in the method is needed.

Padmanabhan et al. proposed *secure traceroute* to protect against the threat that malicious routers could take illegal action against traceroute packets [15]. In this method, a key shared between two routers is used to encrypt a signature. As a result, a limited number of routers can distinguish whether the packet is from traceroute. The disadvantages of this method are the increased number of key processing tasks and the additional memory consumption of the router caused by the increased number of router pairs. Because the method has not yet been implemented, the performance of the system has not been evaluated.

3.2 Prior Inspection

In Internet communication, routes are not constant. To balance the load or control bandwidth on the network, many packets that comprise one communication pursue different routes. Hence, we must take into account the problem that is created when the trusted route is replaced by another route for actual communication. In order to avoid this problem in our method, it is necessary to assume that a large-scale change in the route does not occur within a few minutes. Hence, we investigated the amount of route change by conducting fixed-point observations using tcptraceroute.

From the global IP address of our lab, we executed tcptraceroute for 10 domains randomly chosen from the Alexa list of top 1,000 domains. The experiment was performed over two sessions, and we analyzed the results using the number of unique IP and AS routes, as well as the "Levenshtein distances" of these IP and AS routes.

The "number of unique paths" in the results indicates the number of all routes between the client and a particular host, excluding overlap routes. The Levenshtein distance refers to the degree of similarity between two character strings and is the minimum number of edits (deletions, substitutions, or insertions) needed to change one character string into another. In our evaluation, we replaced "one character" with "one IP address" or "one AS number" and calculated the Levenshtein distance between the nth and $n + 1$th execution, and compared the degree of route similarity. **Table 1** shows the results. The number of IP routes is smaller than eight, and hence, we conclude that it is possible to reconstruct routes using the PPM method. In terms of the number of AS routes, the second score is twice as big as the first score; however, both of these scores are within a permissible range. In addition, the Levenshtein distance of the IP and AS routes is three and two, respectively, and hence, we conclude they are sufficiently small for this method. Given these results, we conclude that route changes over a short time will have an insignificant impact on our system.

Table 1 Results of fixed point observations.

	1st	2nd
IP: Number of Routes	6.4	7.9
AS: Number of Routes	2.5	5.1
IP: Levenshtein Distance	3.26	2.97
AS: Levenshtein Distance	1.63	1.76

3.3 System Specifications

We describe about the specifications of proposed method, especially the marking data and packets authentication.

3.3.1 Marking Data and Construction

The marking data and marking area are set as shown in **Fig. 3**, expanding the method proposed by Savage et al. [5]. A 128-bit data field is generated. This data is divided into two part: 80-bit bit interleaving part and 48-bit hash value. A 80-bit bit interleaving part is composed of a 32-bit IP address, 32-bit AS number, and 16-bit extra data for future extension. This 128-bit field is then divided into eight fragments and stored in a router.

Five-bit data (*initTTL*) is used in this method. In the method proposed by Savage et al., the XOR operation reconstructs the contiguity between two PPM routers. However, if there is a non-PPM router between them, the victim cannot detect it. Assuming a realistic network environment of a mixture of PPM and non-PPM routers, detecting the number and approximate positions of non-PPM routers is required for an accurate reliability evaluation. Thus, by restoring the value of TTL to *initTTL* when a router marks a packet, the difference between *initTTL* and TTL at packet arrival enables the client to determine the distance to a non-PPM router. Eventually, the router marks a packet with a 16-bit fragment, 3 bits for the offset of the fragment and 5 bits for the *initTTL* in the 8-bit Type of Service field, and the 16-bit Identification field in the IPv4 header.

The marking method follows that applied in Goodrich's study [6]. Each router decides whether to make a mark or not based on static probability p and does not XOR the information of two routers, as in the original PPM.

3.3.2 Marking for Specific Communication

In conventional PPM methods, each router marks every packet that passes through it. If the routers did the same in our method, the IP addresses of routers would be carelessly revealed. It is possible that an attacker would exploit an opportunity to attack a network if this system were used. Therefore, in our proposed method, it is necessary to perform marking only for communication between a trusted client and a trusted server. We therefore devised a mechanism that uses a public key cryptographic scheme and performs packet authentication in a supported router. **Figure 4** shows the flow with numbers referring to the entities in the process described in the following. First, a client sends a route detection request to a server (1). The server generates a *Start Kick Packet* (*SKP*) containing the embedded signature of the IP addresses of both the server and client with the server's sign key

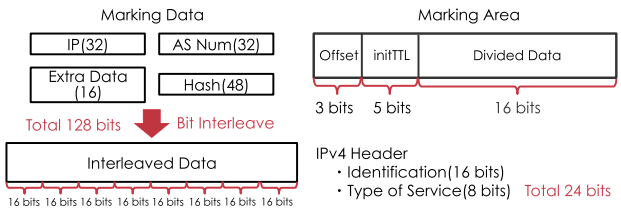


Fig. 3 Marking data and marking area.

© 2015 Information Processing Society of Japan

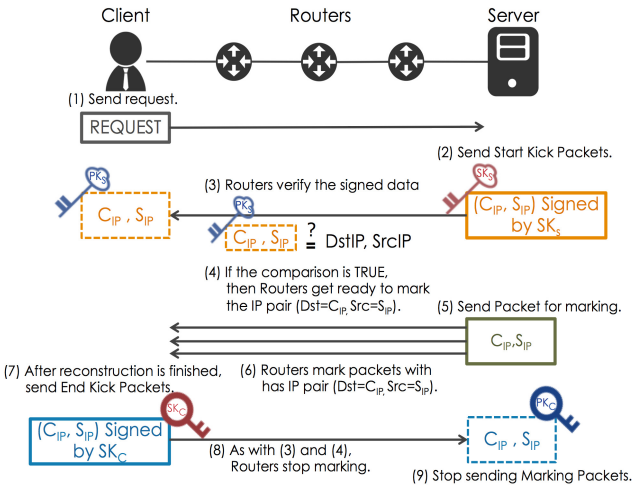


Fig. 4 Identification and marking for a specific communication.

(private key) in its data area and sends it to the client with static trigger bits set in its identification fields (2). When the supported router that relays packets on a route detects the trigger bits, it obtains the server's verify key (public key) from the registry (such as Resource Public Key Infrastructure: RPKI) and uses it to verify the signed packet data (3). If the pair of IP addresses that have been obtained from the signature data are consistent with the source and destination IP on the packet header (4), the router stores this IP pair as a marking target IP pair. Next, the server sends the packets for marking to the client (5), and each router marks the packet if its IP pair equals the target IP pair (6). After the client has gathered sufficient packets to reconstruct the route, it sends the server an *End Kick Packet* (*EKP*) in which an IP pair is signed with the client's sign key (7). Routers stop marking this IP pair (8) exactly as in step (3). When the server receives an *EKP*, it stops sending packets for marking (9).

4. Trust Level Estimation

In this section, we discuss "What is trust in network communication?" or "How should we deal with Trust Level?"

4.1 Concept of Trust Level

The word *trust* has an extensive range of meaning according to the context in which it is used, and difficulties in defining a consistent meaning have been indicated [21]. Jøsang defined *Reliability trust* as follows [22]. "Trust is the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends," citing Gambetta's definition [23]. We can interpret their definition of *trust* as the probability that the demand of a certain entity will be satisfied by another certain entity. Moreover, they stated that one of four elements in *Decision trust*, which can express a broader meaning of trust than *Reliability trust*, is "reliability." We cannot exclude the word *reliability*, since this is the most meaningful concept in the network communication context. *Reliability* in the network communication context is one of the elements in RAS (Reliability, Availability, Serviceability). This means the fault-tolerance of network devices and is quantified by the barometer Mean Time Between Failure (MTBF). We can say that this concept is one of the elements of trust, because MTBF indicates the probability that a user's demand of Layers 1 to 3 in the OSI reference model is satisfied. In the OSI reference model, TCP in Layer 4 or encryption techniques in Layers 5 to 7 can also be interpreted as trust: the demand that packets are sent absolutely by TCP and the demand that the communicated data are preserved by encryption techniques.

The trust model described above using the OSI reference model constitutes only a demand made of each network device or respective communication. In addition, we wish to expand the idea of trust into the social layer. In brief, we want to provide trust information about a social network entity, that is, an AS, for network users. To achieve this, we have to evaluate the communication behavior of the AS, which is determined by the network's communication policy. Some examples of parameters according to the AS's policy are described in the next subsection. Eventually, we want to evaluate the "trust" of an AS using a quantitative value *Trust Level*. Thus, in this paper, we define *Trust Level* as: a quantitative value that evaluates the network communication policy of a certain organization according to a social meaningful barometer.

4.2 Parameters of Trust Level

We propose some useful elements for calculating the *Trust Level* as defined above, as follows. To identify the *Trust Level*, various elements should be used in combination. Then, a radar chart, for instance, can be used to show the results of the calculation of the *Trust Level*. However, in the proposed method we retain only some of the barometer values. We will discuss a strict calculation method as future work.

- AS basic information:
  This includes information, such as management entity, nationality, and the period of fund management.
- Link information between ASs:
  The number of links with other ASs and connection forms (peer, transit) can be utilized by using a scoring method. In Section 6, we construct a simple prototype of an AS link-scoring *Trust Level* scheme.
- Reputation:
  Reputation is considered to be closely related to trust, and is applied to various services [22]. We can use not only a third party reputation service, but also a new reputation system based on that of network entities according to the user.
- Internet Routing Registry (IRR):
  IRR is a database service that stores BGP route information and routing priorities. It is managed by a regional Internet registry or some research institution. Information, such as the presence or absence of admission to IRR or the degree of priority, is an important barometer.
- Management state of AS:
  The management information of an organization, such as financial status, social position, and international status, is an important factor for evaluating the organization. One method for investigating this is to use Web scraping or a Web crawler to monitor Web sites. The *Trust Level* can then be changed according to news stories about organizations.

© 2015 Information Processing Society of Japan

5. Discussion of Security

In this section, we construct and inspect an attack model against our proposed method, and then, we discuss the countermeasures.

5.1 Attack Model

In order to inspect the attack model, we must reconfirm the capabilities of the adversary. Addressing security under an active adversary model enables us to indicate the safety of our proposed method. Therefore, the supposed adversary model is not a semi-honest one, which peeks at the data but does not take any action, but an active model that attacks aggressively.

The purpose of the adversary is to disrupt route detection and trust level estimation. By attacking one of its distinctive functions, the adversary tries to disrupt the process of the proposed method.

Attack models are described in **Table 2**. According to this table, we consider the attack caused by two kinds of adversary: a malicious client and router.

5.1.1 Trigger DoS Attack

In our proposed process, we use two characteristic packets, the *SKP* and *EKP*, as a trigger in order to start key processing. These packets contain a static trigger bit in their packet header so that supported routers recognize them as a "Kick Packet." When the malicious client sends packets with trigger bits to the Internet, supported routers are forced to perform key processing. This causes an increase in the load on the applied router.

To avoid this trigger DoS attack, a threshold is available. When supported routers receive a certain amount of packets, they stop key processing and mitigate their load.

5.1.2 Invalid Request

In the process of our method, we assume the clients and servers model. It is possible that a server will reply to the invalid route detection request from a malicious client.

The countermeasure for this problem is that the server provides user authentication. In the sending request process described in Fig. 1, an appropriate authentication mechanism that is compatible with the server's service would enable it to avoid illegal requests.

5.1.3 Overwriting Attack

When a malicious router overwrites the marking data, which contains divided fragments, their offset, and *initTTL*, or overwrites the header information, the proposed method is influenced. This kind of attack is divided into three types in our method: (1) Fragment overwriting, (2) Distance information tampering, (3) Increasing reconstruct combination attack.

First, we consider fragment overwriting (1). In this attack, a malicious router makes a client reconstruct a different route by overwriting marked data in packets. In **Fig. 5**, when the malicious router observes the packets that are marked by routers *C* and *D*, he overwrites the packets with fragment *X* or *Y*, which have been readied beforehand according to the offset number. It is not necessary to overwrite the distance value in this kind of attack. The gray routers in the figure are the unsupported routers (UR), as mentioned in Section 2. As a result of this attack, the client reconstructs this route as (A, UR, B, Adversary, UR, X, UR, Y), as the malicious router intended, as opposed to the correct route (A, UR, B, Adversary, UR, C, UR, D). To prevent this attack, we consider that investing marking data with a digital signature is a good solution. Supported routers register their public key with a PKI (Public Key Infrastructure) such as RPKI, and then, the client verifies the signature using the router's public key when he reconstructs. This leads to an increase in the client load; however, it prevents a malicious router from creating illegal marking data.

Table 2 Attack model.

Adversary	Attack
Client	Trigger DoS Attack Invalid Request
Router	Overwriting Attack Packet Blackhole

Now, we consider distance information tampering (2). Distance information indicates the hop number from a client to a supported/unsupported router and is calculated by the difference between TTL and *initTTL*. A malicious router can influence the reconstruct process by manipulating the TTL or *initTTL* value within the condition $TTL \leq initTTL$. As **Fig. 6** shows, a client identifies incorrectly that the distance from the adversary to router *C* is 15, by mistakenly estimating that the distance to router *C* is 19 where the correct distance is 6. An advantage of this attack is that it makes a client reconstruct a biased route. However, in this case, the merit of the attack's success is limited: since the client can estimate the distance to the server, he can detect that the received packets are overwrites in some cases.

The third type of attack, increasing reconstruct combination attack (3), increases the number of combinations that are required for verification in the reconstruct process by fragment tampering and multiplying the distance information. However, a client can estimate the number of combinations, because he can estimate the distance to a server from the proposed process. Using this

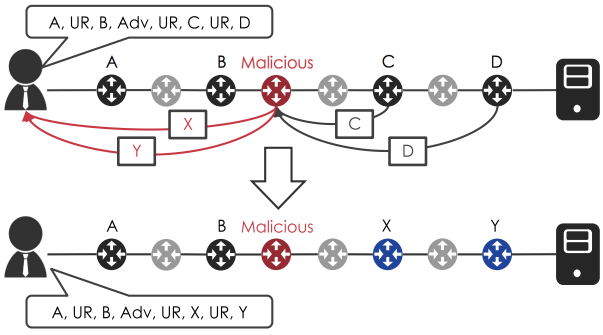


Fig. 5 Fragment overwriting.

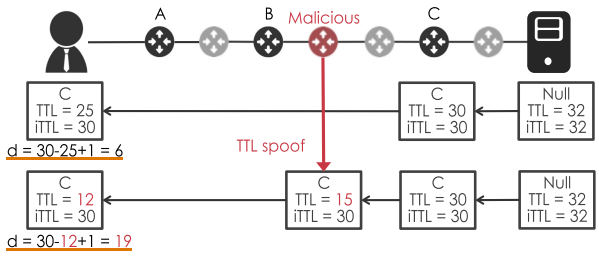


Fig. 6 Distance information tampering.

© 2015 Information Processing Society of Japan

estimation, a client can identify this attack and stop the proposed process by dropping the attack packets.

5.1.4 Packet Blackhole

Packet blackhole is an attack where a malicious router drops packets that are used in the proposed method. It is possible to drop *SKP* and *EKP* by identifying the trigger bit or marking packets that are authenticated between a client and a server. An advantage attained by this attack is that it interferes with the success of the trust level estimation by hiding the upstream route information. This causes a client not to collect sufficient route information. However, if the client sets the lower threshold for the amount of route information, he can decide that this route is not trustworthy because of the number of collected packets. Moreover, if this attack is continued by the same malicious router, in our opinion the adversary can probably be detected by using an existing tool, such as traceroute. Therefore, we conclude that the influence of this attack is limited, and a countermeasure is not required.

6. Implementation and Evaluation

In this section, we show the implementation method and evaluation result.

6.1 Implementation

6.1.1 Outline of Implementation

We implemented two elements: route detection and trust level estimation. The steps implemented in route detection correspond to (1), (2), and (3) in Fig. 1. For trust level estimation, we implemented a prototype of a Trust Level Knowledge Base, (4), (5), and (7). The implementation of the remaining complementary process for route detection (6) remains as future work.

6.1.2 Implementation Method of Route Detection

In this study, we implemented: (1) a Linux kernel with a marking function, (2) a userland application on the router, and (3) a client and server.

First, we developed (1) a Linux kernel in which the proposed method was implemented. In our implementation, packet marking is executed in Layer 2, because we developed the marking method not as a router function but as a bridge function, for simplicity's sake. However, the operations of writing information in the IP header and recalculating the checksum of the header are executed in Layer 3, and therefore, we will be able to expand our implementation as a router function. Instead of embedding the marking probability configuration or various parameters into the kernel, we used *sysfs*, a virtual file system provided for Linux kernel 2.6, and later, set the parameters for the kernel. Userland settings can result in not only flexible marking probability but also flexible marking methods.

Second, we implemented (2) a userland application. So that the communication identification mechanism using a key will operate correctly, the router must be able to decrypt. We implemented decryption by causing the Linux kernel to cooperate with a Java application running in userland. This application retrieves public keys and authenticates packets by decrypting data. A pair of IP addresses that are successfully authenticated and become a marking target are passed to the kernel through *sysfs*. The kernel then adds this pair to a list and uses it for detecting marking target IPs. This Java application was implemented in Java SE-1.6 and executed in Java SE-1.7.

Third, (3) the client and server programs were implemented to execute route detection processes. Moreover, the client software accesses a Trust Level Knowledge Base after route detection. In this element (3), we used Jpcap, a Java library for capturing and sending network packets [25].

6.1.3 Implementation of Prototype Trust Level System

In Section 4, we defined the *Trust Level* but did not determine a strict method for calculating it. In order to evaluate the proposed method, especially total running time of proposed system, we require a simple scheme for quantifying *Trust Level*. Then, we can easily calculate the *linkscore* based on AS link information, and set this barometer in the database to reply to the query from the client.

First, we obtained the AS number list that contains the organization and nationality of an AS from BGP Reports, which are provided by APNIC [26]. Next, we obtained AS link information from the Internet AS-level Topology Archive [27]. In this information, the link between ASs is classified into four patterns: p2p, c2p, p2c, unknown. We calculated the *linkscore* using

$$linkscore = (p2p/total * 1 + c2p/total * 1.2 + p2c/total * 0.8 + unknown * 0.1)/10 \quad (1)$$

The calculated *linkscore* and AS basic information are stored in MySQL database. The appropriateness of this scoring is not discussed in this paper, since we just want to test the total running time.

6.2 Evaluation

We evaluate the performance of our implementation by measuring three items; the throughput, load of implemented device, and running time of proposed method.

6.2.1 Throughput

To verify the impact on the communication speed caused by implementing the proposed method in a kernel, we measured the throughput of the prototype device.

A packet sender (Server), a packet receiver (Client), and the proposed method implemented on a bridge device (Linux Server Machine) were connected in series (**Fig. 7**). The proposed method devices were implemented in the environment detailed in **Table 3**. The server model was MacBook Pro (Late 2011), which was equipped with a 1 Gbps NIC. The client model was a MacBook Air (mid 2011) with a 1 Gbps Apple Thunderbolt Ethernet adapter.

The following four configurations were measured for comparison:

- Proposed method with $p = 0.5$ (UDP packets)



Fig. 7 Single device topology.

© 2015 Information Processing Society of Japan

Table 3 Implementation environment details.

Machine	IBM x306m
CPU	Intel Pentium 4531 (3.0 GHz)
Memory	2 GB
NIC	Broadcom 5721 (1 Gbps) × 2
Linux Kernel	2.6.32
Linux Distribution	Debian 6.0.1
Java	Java SE Runtime Environment 1.7.0

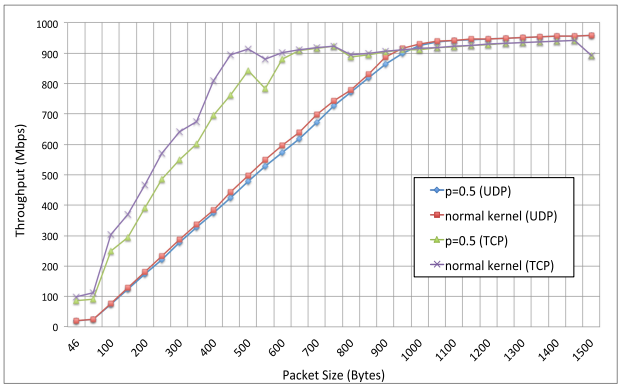


Fig. 8 Throughput for each parameter.

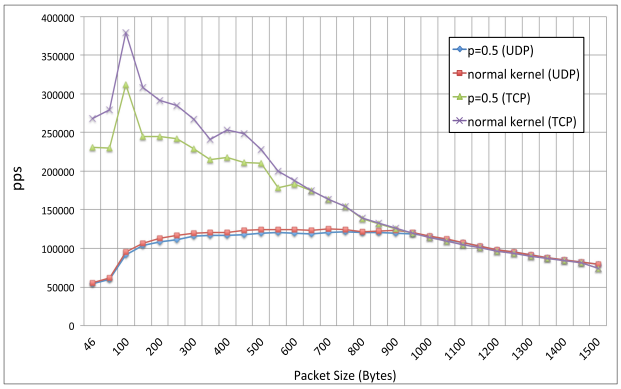


Fig. 9 Rate (pps) for each parameter.

- Normal kernel (UDP packets)
- Proposed method with $p = 0.5$ (TCP packets)
- Normal kernel (TCP packets)

The label "normal kernel" indicates a kernel in which the proposed method was not implemented that simply relays packets as a bridge. In the kernel in which the proposed method was implemented, we set the marking probability p to 0.5 and maximized the number of processes to check whether the received packet was marking the target. Both UDP and TCP were measured using *netperf*.

The throughput of each of the four configurations was measured for several packet sizes ranging from 46 to 1,500 (based on the largest MTU in Ethernet), and was averaged over 10 trials. We then calculated the packets per second (pps) given the throughputs and packet sizes. **Figures 8** and **9** graphically depict our results.

There was not a large difference among the results for UDP. This indicates that the impact of introducing the proposed method into network devices in terms of simple packet forwarding without connections is not significant. As a comparison with the original PPM methods, we mention the results of Okada et al. [11] for the first implementation of PPM. There is a difference in that,

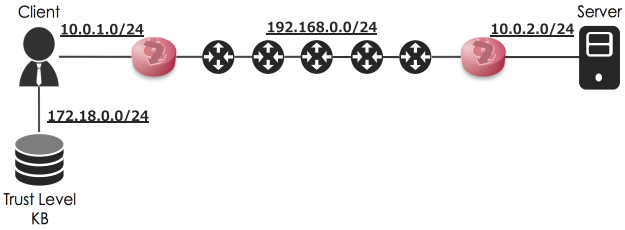


Fig. 10 Multiple devices topology: Serial.

according to Okada's results, the maximum throughput (about 900 Mbps) was reached with a packet size of 400 Bytes. However, we concluded that this disparity is caused by machine performance, because the throughput of the normal kernel is different.

According to Okada's study, the level of marking probability has very little influence on throughput or pps. Therefore, in our method, we set the marking probability freely without needing to consider its effect on communication speed.

The performance of the kernel in which the proposed method was implemented, on the other hand, was inferior to that of a normal kernel when the packet size ranged between 100 and 650 Bytes for TCP. In our opinion, the reason for this result is that TCP makes the connection, initiated by a three-way handshake, and this may cause a delay in the performance of the devices. However, Okada et al. reported that the distribution of packet sizes in real communication can be almost completely divided into two types: very small (about 50 Bytes) and very large (about 1,450 Bytes) [11]. According to this observation, the packet forwarding could operate at a high speed in our proposed method, similar to that of a normal kernel, in a real-world environment.

6.2.2 Load of Proposed Method

We measured the load required to execute the proposed method in a Linux server machine in order to estimate the overhead of our method. In the proposed method, the router (bridge) must retrieve public keys from the key server. Therefore, we implemented a simple key-offering server. However, there is a limited number of NICs in a Linux server machine, and hence, the Client, described in **Fig. 10**, plays the role of key server once. The client and server model was same as those described in Section 6.2.1.

To measure the overhead, we set the marking probability at $p = 0.5$ and executed the proposed method 10 consecutive times and observed the free memory or CPU use rate using the *vmstat* command before and after execution. This sequence was averaged over 10 trials.

The results show that the reduction in free memory after the execution of the proposed method was about 253.2 KBytes and the CPU use rate was only 2.6%. From these observations, we conclude that the load of the proposed method is negligible for the network device.

6.2.3 Running Time

Running time is an important factor when using this type of system. We prepared several identical machines running the proposed method, as described in Table 3, and connected these machines in series (as shown in Fig. 10) and in parallel (as shown in **Fig. 11**). In Fig. 10, we put five supported machines between two routers (Cisco 1812j and Yamaha RTX1100) in series, and

© 2015 Information Processing Society of Japan

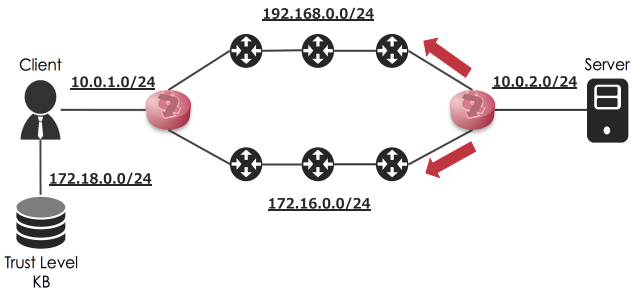


Fig. 11 Multiple devices topology: Parallel.

Table 4 Method running time: Serial.

(milliseconds)	Average	Median	Min	Max
Send Request	9.66	9	8	14
Receive Start Kick Packet	544.54	542.5	523	588
Marking and Reconstruct	3,461.8	3,507.5	2,284	3,848
Send End Kick Packet	16.76	17	12	20
Trust Level Estimation	616.94	615	602	664
Total	4,675	4,715.5	3,522	5,087

Table 5 Method running time: Parallel.

(milliseconds)	Average	Median	Min	Max
Send Request	10.06	10	8	15
Receive Start Kick Packet	541.94	542	471	595
Marking and Reconstruct	3,157.5	3,348.5	1,274	5,412
Send End Kick Packet	16.96	16.5	13	22
Trust Level Estimation	693.3	685	670	779
Total	4,419.7	4,587.5	2,505	6,672

execute the proposed method. As for Fig. 11, we use same two routers to test the proposed method when implemented in devices in multiple routes. The right router distributes packets using two routes (192.168.0.0/24 and 172.16.0.0/24).

The client and server was the same as that in Section 6.2.1. After route detection process, the client accesses the Trust Level Knowledge Base and inquires about the *Trust Level* of the reconstructed routes.

We measured running time over the following five procedures: sending the request, receiving the *SKP*, gathering and reconstructing the marked packets, sending the *EKP*, and *Trust Level* estimation. We repeated these procedures 50 times and calculated the averages. The marking probability was set to $p = 0.082$, which Okada et al. [10] claim is the optimum probability in the Internet topology.

As a result, route detection and *Trust Level* estimation was successfully finished. **Tables 4** and **5** lists our result. The average total running time was 4.6 s in serial and 4.4 s in parallel, sufficiently fast that users do not notice delays. Even in the worst case, the client can obtain route information and judge the trust level within about 6 s. As future work, we have to test the running time in more realistic environment, that have long network distance.

7. Limitation and Future Work

The proposed method achieves a novel function that identifies the route trust level based on information obtained from intermediate routers or ASs. However, it still has limitations in several respects. In this section, the limitations of the proposed method are outlined. Future work to address these limitations is then discussed.

7.1 Proposed Method for IPv6

In this study, IPv4 was assumed. However, IPv6 should be also considered. In the proposed method, the overall scheme does not depend on the IP version. We need to consider only the version of PPM used. Although IPv6-ready PPM is outside the scope of this paper, there are several other papers that discuss IPv6-ready PPM [28], [29]. If there is a lightweight PPM implemented on IPv6, it could be applied to the proposed method.

7.2 Deployability

When we consider a novel method, such as adding some functionality to network equipment, deployability is always a problem. In the work of Okada et al., a device for applying the PPM function to working routers was proposed [11]. This could also be applied to the proposed method. Evolving Network Functions Virtualization (NFV) might also accelerate deployability.

7.3 Bidirectionality

The proposed method focuses on the route information returning from servers, that is, in a single direction from the server to client. Reverse traceback has a purpose that is similar to that of the proposed method. A case could occur where the opposite direction is also important. For such a case, bidirectionality should be considered.

To achieve bidirectionality, essentially, the solution is to deploy another system in the opposite direction, although it might cause a flow concentration problem for a server with many clients.

7.4 Route Information Complementation

The proposed method achieves the objective of obtaining route information, but the possibility of incomplete route detection still remains. Merging information of the results of our proposed method and BGP tables might be a convincing method of complementing route detection.

8. Conclusion

In this paper, a method for detecting the route and identifying route trust level between a client and a server was proposed. To identify the trust level, we use PPM, proposed packet authentication, and knowledge bases from trusted third parties. A prototype system of the proposed method was developed and evaluated, and the results show its feasibility.

To the best of our knowledge, the proposed method is the first that identifies route trust level based on information obtained from intermediate routers or ASs.

References

[1] The Guardian: NSA Prism program taps in to user data of Apple, Google and others, available from ⟨http://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data⟩ (accessed 2014-11-10).

[2] The Washington Post: U.S., British intelligence mining data from nine U.S. Internet companies in broad secret program, available from ⟨http://www.washingtonpost.com/investigations/us-intelligence-mining-data-from-nine-us-internet-companies-in-broad-secret-program/2013/06/06/3a0c0da8-cebf-11e2-8845-d970ccb04497_story_1.html⟩ (accessed 2013-11-10).

© 2015 Information Processing Society of Japan

[3] Reuters: Exclusive: Secret contract tied NSA and security industry pioneer, available from ⟨http://www.reuters.com/article/2013/12/20/us-usa-security-rsa-idUSBRE9BJ1C220131220⟩ (accessed 2013-12-06) (Dec. 20, 2013).

[4] InfoWorld: Snowden: The NSA planted backdoors in Cisco products, available from ⟨http://www.infoworld.com/d/the-industry-standard/snowden-the-nsa-planted-backdoors-in-cisco-products-242534⟩ (accessed 2014-05-15).

[5] Savage, S., Wetherall, D., Karlin, A. and Anderson, T.: Practical network support for IP Traceback, *Proc. ACM SIGCOMM '00*, pp.295–306 (2000).

[6] Goodrich, M.T.: Probabilistic Packet Marking for Large-Scale IP Traceback, *IEEE/ACM Trans. Networking*, Vol.16, No.1, pp.15–24 (2008).

[7] Song, D. and Perrig, A.: Advanced and Authenticated Marking Schemes for IP Traceback, *Proc. IEEE INFOCOM*, pp.876–886 (2001).

[8] Dean, D., Franklin, M. and Stubblefield, A.: An Algebraic Approach to IP Traceback, *Proc. Network and Distributed System Security Symp.* (*NDSS*), pp.3–12 (2001).

[9] Law, T.K.T., Yau, D.K.Y. and Lui, J.C.S.: You can run, but you can't hide: An effective statistical methodology to trace back DDoS attackers, *IEEE Trans. Parallel Distrib. Syst.*, Vol.16, No.9, pp.799–813 (2005).

[10] Okada, M., Kanaoka, A., Katsuno, M. and Okamoto, E.: Probability Estimation for Probabilistic Packet Marking, *IPSJ Journal*, Vol.52, No.9, pp.2718–2728 (2011).

[11] Okada, M., Goto, N., Kanaoka, A. and Okamoto, E.: A Device for Transparent Probabilistic Packet Marking, *Proc. 4th IEEE International Workshop on Network Technologies for Security, Administration and Protection* (*NETSAP2013*), pp.242–247 (2013).

[12] Liu, J., Lee, Z.-J. and Chung, Y.-C.: Dynamic probabilistic packet marking for efficient IP traceback, *Computer Networks*, Vol.51, No.3, pp.866–882 (2007).

[13] Jacobson, V.: Traceroute, available from ⟨ftp://ftp.ee.lbl.gov/traceroute.tar.gz⟩ (accessed 2014-05-20).

[14] Michael, T.C.: Tcptraceroute: An implementation of trace route using TCP SYN packets, available from ⟨http://michael.toren.net/code/tcptraceroute/⟩ (accessed 2014-05-20).

[15] Padmanabhan, V.N. and Simon, D.R.: Secure Traceroute to Detect Faulty or Malicious Routing, *ACM SIGCOMM Computer Communications Review*, Vol.33, No.1, pp.77–82 (2003).

[16] Katz-Bassett, E., Madhyastha, H., Adhikari, V., Krishnamurthy, A. and Anderson, T.: Reverse traceroute, *Proc. 7th USENIX Conference on Networked Systems Design and Implementation*, pp.15–15 (2010).

[17] Magoni, D. and Hoerdt, M.: Internet core topology mapping and analysis, *Computer Communications*, Vol.23, No.5, pp.494–506 (2005).

[18] Govindan, R. and Paxson, V.: Estimating Router ICMP Generation Delays, *Proc. Passive and Active Measurement Workshop* (*PAM*), Fort Collins, CO (2002).

[19] Marchetta, P., Persico, V., Katz-Bassett, E. and Pescape, A.: Don't trust traceroute (completely), *Proc. ACM Conext Student Workshop*, pp.5–8 (2013).

[20] Alexa: Top 1,000,000 Sites, available from ⟨http://www.alexa.com/⟩ (accessed 2013-08-01).

[21] McKnight, D.H. and Chervany, N.L.: The Meanings of Trust, Technical Report MISRC Working Paper Series 96-04, University of Minnesota, Management Information Systems Research Center (1996).

[22] Jøsang, A., Ismail, R. and Boyd, C.: A survey of trust and reputation systems for online service provision, *Decision Support Systems*, Vol.43, No.2, pp.618–644 (2007).

[23] Gambetta, D.: Can we trust trust?, *Trust: Making and Breaking Cooperative Relations*, Gambetta, D. (Ed.), pp.213–238, Basil Blackwell, Oxford (1990).

[24] Merit Network: Internet Routing Registry, available from ⟨http://www.irr.net/⟩ (accessed 2014-11-13).

[25] Fujii, K.: Jpcap, available from ⟨http://netresearch.ics.uci.edu/kfujii/Jpcap/doc/index.html⟩ (accessed 2014-05-10).

[26] AS Names: BGP Reports, available from ⟨http://bgp.potaroo.net/cidr/autnums.html⟩ (accessed 2014-11-14).

[27] Zhang, Y.: IRL AS-level Topology Archive, available from ⟨http://irl.cs.ucla.edu/topology/ipv4/relationship/⟩ (accessed 2014-11-14).

[28] Dang, X., Albrighe, E. and Abonamah, A.A.: Perfomance analysis of probabilistic packet marking in IPv6, *Computer Communications*, Vol.30, No.16, pp.3193–3202 (2007).

[29] Tripathy, A., Dansana, J. and Mishara, D.P.: A secure packet marking scheme for IP traceback in IPv6, *Proc. International Conference on Advances in Computing, Communications and Informatics* (*ICACCI '12*), pp.656–659 (2012).



Nasato Goto was a master student at University of Tsukuba. He received his B.E. degree from University of Tsukuba in 2013. His research interests is network security.



Akira Kanaoka received his Ph.D. degree in engineering from University of Tsukuba, Japan in 2004. He worked at SECOM Co., Ltd. from 2004 to 2007, and at University of Tsukuba from 2007 to 2013. He is currently an assistant professor of Department of Information Science, Faculty of Science, Toho University. His research interests include network security and cryptographic application.



Masayuki Okada works in the Engineering Department at JPNIC. He experienced a BGP operation of an academic network since 2000. Mr. Okada joined JPNIC in 2004 and is responsible for the development and operation of the IP resource management system related to routing and JPIRR research, as well as the use of IRR. In recent years, he has focused his efforts on outreach about RPKI technology and its operational deployment. He finished his Ph.D. in 2012 in Computer Science.



Eiji Okamoto received his B.S., M.S. and Ph.D. degrees in electronics engineering from Tokyo Institute of Technology in 1973, 1975, 1978, respectively. He worked and studied communication theory and cryptography for NEC central research laboratories since 1978. In 1991 he became a professor at Japan Advanced Institute of Science and Technology, then at Toho University. Now he is a professor at Faculty of Engineering, Information and Systems, University of Tsukuba. His research interests are cryptography and information security. He is a coeditor-in-chief of International Journal of Information Security and a member of IEEE and ACM.

© 2015 Information Processing Society of Japan