Detecting and Understanding Online Advertising Fraud in the Wild*

Fumihiko KANEI^{†,††a)}, *Nonmember*, Daiki CHIBA[†], Kunio HATO[†], Katsunari YOSHIOKA^{††},
Tsutomu MATSUMOTO^{††}, and Mitsuaki AKIYAMA[†], *Members*

SUMMARY While the online advertisement is widely used on the web and on mobile applications, the monetary damages by advertising frauds (ad frauds) have become a severe problem. Countermeasures against ad frauds are evaded since they rely on noticeable features (e.g., burstiness of ad requests) that attackers can easily change. We propose an ad-fraud-detection method that leverages robust features against attacker evasion. We designed novel features on the basis of the statistics observed in an ad network calculated from a large amount of ad requests from legitimate users, such as the popularity of publisher websites and the tendencies of client environments. We assume that attackers cannot know of or manipulate these statistics and that features extracted from fraudulent ad requests tend to be outliers. These features are used to construct a machine-learning model for detecting fraudulent ad requests. We evaluated our proposed method by using ad-request logs observed within an actual ad network. The results revealed that our designed features improved the recall rate by 10% and had about 100,000–160,000 fewer false negatives per day than conventional features based on the burstiness of ad requests. In addition, by evaluating detection performance with long-term dataset, we confirmed that the proposed method is robust against performance degradation over time. Finally, we applied our proposed method to a large dataset constructed on an ad network and found several characteristics of the latest ad frauds in the wild, for example, a large amount of fraudulent ad requests is sent from cloud servers.

key words: online advertising, advertising fraud, adware

1. Introduction

As online advertising is one of the essential monetization models on the web and mobile applications, its market size has been increasing rapidly over the last several years. The Interactive Advertising Bureau (IAB) reported that online advertising revenues in the United States totaled \$107.5 billion for the full year of 2018 [2]. As the online advertising market has grown, the damage caused by advertising frauds (ad frauds), such as automated clicks of advertisements (ads) by bots, is also increasing. The IAB reported that fraudulent activities cost the U.S. digital advertising industry \$8.2 billion annually [3].

Over the past decade, researchers have been investigating the mechanism of ad frauds by analyzing network traffic

originating from online ads [4], [5] and client applications (e.g., browser extensions and mobile applications) that display ads to users [6], [7]. Some researchers also focused on detecting ad frauds. They proposed anomaly detection of click-spam [8], [9] and correlation analysis for detecting crowd-sourced ad frauds [10].

There are various types of ad frauds in the wild. The most sophisticated attack is a distributed attack that abuses many clients and publisher websites. In this case, malicious activities launched from various clients and publisher websites are blended into legitimate activities. This type of attack cannot be detected using a simple anomaly-based method that leverages the burstiness of ad requests, thus such a method causes non-negligible false positives and false negatives.

We propose a method for precisely detecting fraudulent ad requests observed within an ad network. The proposed method is a machine-learning-based method that leverages features extracted from ad requests and classifies them as either benign or malicious. To design features robust against attacker evasion, we introduce the assumption that attackers cannot know of the statistics of ad requests from legitimate users because they cannot observe ad requests from the standpoint of the ad network. On the basis of this assumption, we statistically calculate the feature values from both client and publisher information and use them to distinguish ad frauds from a large amount of legitimate ad requests.

We make the following contributions.

- We developed a method for precisely detecting ad requests related to fraudulent activities observed within an ad network. Our proposed method leverages features that are difficult to control from the standpoint of attackers and are robust against evasion attacks.
- The results of an evaluation reveal that our proposed method can effectively detect ad frauds and our features improve the recall rate by 10% compared with conventional features based on the burstiness of fraudulent ad requests.
- From the evaluation involving a long-term dataset, we confirm that our proposed method is more robust against performance degradation over time compared with a conventional method.
- We applied our method to a large ad-request dataset collected within an actual ad network and empirically analyzed *in-the-wild* fraudulent activities targeting on-

Manuscript received August 23, 2019.

Manuscript revised December 25, 2019.

Manuscript publicized March 24, 2020.

[†]The authors are with NTT Secure Platform Laboratories, Musashino-shi, 180–8585 Japan.

^{††}The authors are with Yokohama National University, Yokohama-shi, 240–8501 Japan.

*This paper is the extended version of the paper presented at COMPSAC 2019 [1].

a) E-mail: fumihiko.kanei.sw@hco.ntt.co.jp

DOI: 10.1587/transinf.2019ICP0008

line ads. Our analysis revealed that about 6.9% of over 8 million impressions are related to ad frauds.

- We reveal several properties of the latest ad frauds in the wild. For example, a large amount of fraudulent ad requests originates from IP addresses managed by cloud-service providers, and attackers tend to use (or pretend to use) a specific OS version to carry out ad frauds.

The rest of this paper is organized as follows. In Sect. 2, we present the background of online ads and the threat models of ad fraud. Section 3 summarizes related work. In Sect. 4, we describe our method of detecting ad frauds. In Sect. 5, we discuss the evaluation of our method's effectiveness in comparison with current methods. We also present the results from empirical analysis of ad fraud by applying our method to a large dataset. In Sect. 7, we discuss the limitations of our method, future work, and the ethics and privacy in this research. We conclude the paper in Sect. 8.

2. Background

We explain the mechanism of online advertisement and the threat models of ad fraud as the research background.

2.1 Online Advertising

The main stakeholders of online advertising are end users, publishers, advertisers, and ad networks. Figure 1 shows the flow of ad placement when a user accesses a website via a browser. First, an end user accesses a publisher's website that contains ads (Step 1). The response from the publisher's website containing ad-tags (typically `<script>` or `<iframe>` tags) is loaded on the user's browser (Step 2). Next, JavaScript code loaded by an ad-tag is executed on the browser, and an ad request is sent to an ad server managed by an ad network provider (Step 3). Information about the user and publisher is included in the ad request, such as the category of the publisher's website, size of ad space, user's browser, operating system (OS) version, and cookies. Then, ad bidding is conducted in the ad network (Step 4). Ad bidding is an auction-like process that determines which ad will be displayed on a user's browser. In the bidding process, the ad network mediates between the publisher and advertiser by using information included in the ad request. After the ad bidding, the information about the bidding winner is returned, and an ad is displayed on the browser (Step 5). When a browser loads ad content such as a banner image, an analytics script is also loaded for measuring the effectiveness of the ad. Finally, the user is redirected to an advertiser's website when she/he clicks the ad (Step 6).

The advertising cost charged to an advertiser is determined using various metrics. Typically, the cost of impressions is determined in accordance with Cost Per Mille (CPM), which is calculated per thousand impressions by end users. Similarly, the advertiser is charged in accordance with Cost Per Click (CPC), which is determined per click.

Fig. 1 Flow of online ad placement

Cost Per Action (CPA) is another metric that is charged per user's action (e.g., buying a product on an e-commerce site) after an advertisement is clicked.

2.2 Threat Model

An ad fraud is an attack that defrauds advertisers of advertising funds. Online ads are usually placed with the aim of reaching human audiences such as end users accessing advertisers' websites. On the other hand, attackers generate invalid web traffic. This type of attack, known as impression spam, produces a large amount of meaningless ad impressions. Similarly, attackers also generate programmatic clicks without the intent of the end users, which is called click spam.

Attackers inflate impressions and clicks in various ways. The strategy of attacks is roughly classified into two groups: attacks by humans and attacks by programs. The former is manually launched by people recruited by an attacker. For example, an attacker can leverage crowdsourcing to hire workers at low cost and direct them to access a specific website and click ads [10]. An attacker can also abuse the activities of legitimate users by tricking them on the publisher's website [11]. The latter launches ad frauds in a programmatic manner to automate ad requests. For example, attackers can execute an auto-clicker program on their machines. Attackers can also generate automated ad requests by using machines infected with malware, adware, and PUPs (Potentially Unwanted Programs) [12], [13].

In addition, a sophisticated attacker conducts distributed attacks to evade ad-fraud detection. Figure 2 shows an overview of distributed ad frauds. An attacker prepares an infrastructure that consists of many clients and websites that are controlled by the attacker. Then, the clients access the websites with limited throttle. These requests can be sent from many machines (e.g., botnets), or few machines impersonating various client environments (e.g., manipulating cookies and user agents, using HTTP proxies). This type of attack is difficult to detect because the amount of ad requests for each client and website managed by an attacker does not differ from that for legitimate ones.

Fig. 2 Overview of distributed ad fraud

3. Related Work

Security research conducted on online advertisement analysis has focused on understanding the ecosystem of malicious ads and detecting ad-based threats.

To understand the mechanisms of ad fraud and characterize ad fraud precisely, many studies have been conducted on analyzing *client-side* programs: click bots [12]–[14] and malicious ad extensions [15], [16].

Previous studies on detecting ad fraud were conducted on the basis of the underlying properties of the threat: burstiness of requests [11], [17], [18], coalition activity (uniformity of behavior) [19], [20], anomaly of HTTP request sequence [6], high click-through rate [11], and high revenue of investment [9], [11]. Efficient algorithms based on bloom filters for detecting *duplicate* clicks were proposed [17], [18]. Metwally et al. and Yu et al. focused on coalition activity, which is a sophisticated ad-fraud technique [19], [20]. Dave et al. proposed a method of detecting publishers with anomalously high return on investment (ROI) by using ad-click logs including the publisher-user pair and revenue for each click on the basis of an assumption that a click-spam attack must have higher ROI for the click-spammer than an ethical publisher [9]. Stone-Gross et al. proposed a method combining various properties [11]. Its unique property is the click-through rate of ads that relies on an assumption that a malicious client has a higher click-through rate than an innocent client. The main difference between the above studies and ours is the fundamental property of attackers used to distinguish fraudulent activities from legitimate ones. In this study, we utilized the statistical knowledge of legitimate users that cannot be known by attackers.

Whereas many studies have focused on observation from the perspective of the end user or gateway of a local network, a small number of studies including ours have attempted to detect ad fraud from the *server-side* perspective [5], [9], [11]. In these studies, the following types of ad traffic were analyzed: the vantage point of a demand site platform (DSP) between ad exchanges and advertis-

ers [5], ingress and egress ad traffic from a specific ad exchange [11], and ad traffic between publishers and ad network [9]. Server-side observation has a strong advantage; it collects a large amount of ad traffic at the vantage point of online advertisement. Although the above server-side observations exhibit scalable observation, some information about clients is sometimes not available due to the lack of transparency for considering security and privacy, e.g., IP addresses were masked or hashed in previous research [5], [11]. The observatory that construct our dataset was also located at the server-side, which is the point between publishers and ad networks, like that of Dave et al [9]. Our dataset retained not only basic information of impression and click events (e.g., timestamps and IP addresses) but also end user environments and their behavioral information.

4. Proposed Method

In this section, we describe our proposed method for detecting ad frauds. As static rule-based detection (e.g., extraction of malformed ad request) does not scale because writing detection rules requires manual effort, anomaly-based detection based on the burstiness of ad requests has been recently studied. However, as mentioned in Sect. 2, an anomaly-based method based on the amount of ad requests cannot detect distributed ad frauds: it causes non-negligible false positives and false negatives because an attacker can easily change the amount of ad requests.

Considering the above, we propose a method for precisely detecting ad frauds. We introduced the following assumptions and developed the method on the basis of them: *an attacker cannot observe a large amount of ad requests reaching an ad network or know of the statistics about the client environment and publisher website of the legitimate users*. In particular, we calculate the appearance frequency of each client environment, publisher website, and combination of them by using ad-request logs observed within an ad network. Because an attacker cannot know of this information, ad requests from the clients and publisher websites controlled by an attacker must have tendencies different from ad requests from legitimate ones. We constructed feature vectors including the above statistics and built a model to classify ad requests as either benign or malicious. Our proposed method was designed for detecting ad frauds in co-operation with an ad network. Therefore, a malicious stakeholder within an ad network, which may have the capability to monitor a large amount of ad requests from legitimate users, is beyond the scope of this paper. The rest of this section describes the details of the proposed method.

4.1 Overview

Figure 3 shows an overview of the proposed method. Our proposed method is composed of two phases: training and classification. The input of the training phase is a series of ad-request logs with ground truth labels (i.e., fraud or non-

Table 1 Features used to detect fraudulent ad requests. Conventional features were developed on the basis of prior research [11]

	No.	Feature name	Description	Feature type	Source of feature
Newly designed features	1	rDNS-e2LD_freq	Appearance frequency of e2LD extracted from rDNS of client IP address	Numeric	Client
	2	FQDN_freq	Appearance frequency of FQDN extracted from publisher URL	Numeric	Publisher
	3	rDNS-e2LD and FQDN	Combinational appearance frequency of rDNS-e2LD and FQDN	Numeric	Client & publisher
	4	geoip	Geolocation of client IP address	Categorical	Client
	5	dynamic_range	Whether client IP address is in dynamic ranges	Boolean	Client
	6	alexa_rank	Alexa Rank of FQDN extracted from Publisher URL	Numeric	Publisher
Conventional features	7	impression_count	Amount of impressions from client	Numeric	Client
	8	ctr	Click-through rate	Numeric	Client

Fig. 3 Overview of proposed method

fraud). A time point t is defined, and ad requests observed in the ad network during the previous Δt_1 hours from t are used for the training dataset. In the training phase, we extract features from ad-request logs and train a machine-learning model. The input of the classification phase is the trained model constructed in the training phase and a series of ad-request logs observed between t and $t + \Delta t_2$. The final output of our method is the classification result of each ad-request log tested in the classification phase.

4.2 Feature Extraction

We extracted eight features from ad requests observed in an ad network. Table 1 lists the features used with the proposed method. Features Nos.1–6 are newly designed in this study, and the others were proposed in prior research [11]. We explain the details of each feature in the rest of this section.

Newly designed Features: Features Nos.1–3 listed in Table 1 are calculated on the basis of the appearance frequency of the client environment and publisher’s website among all ad requests observed within the ad network. Figure 4 shows the procedure of feature calculation. First, we extract the following two values from the training dataset and count the number of appearances of each unique value.

- **rDNS-e2LD_freq:** We calculate the appearance frequency of an effective second-level domain (e2LD) extracted from the reverse DNS (rDNS) lookup result of the client IP address. An e2LD is the smallest unit of a domain name that can be registered by Internet users. For example, when the rDNS is `foo1.example.co.jp`, the e2LD is `example.co.jp`. The e2LD part can be extracted from any domain name

by using the Public Suffix List [21]. In general, the IP addresses managed by the same organization for a specific purpose (e.g., dynamic assignment by an Internet Service Provider (ISP), a company’s internal networks, etc.) tend to have a common e2LD in rDNS (rDNS-e2LD). We use an rDNS-e2LD rather than an IP address as a feature representing a client. We assume an rDNS-e2LD is more stable than an IP address because an rDNS-e2LD is generally more costly to change than an IP address.

- **FQDN_freq:** We calculate the appearance frequency of a fully qualified domain name (FQDN) extracted from publisher URLs. We use the FQDN rather than URL as a feature representing the publisher, because it is costly for an attacker to prepare a new FQDN and register it for a publisher website.

We also calculate the combinational appearance frequency of the above two features: **rDNS-e2LD and FQDN**. We list all pairs of unique values for these two features and count the number of appearances of each pair in the training dataset. The tables on the right in Fig. 4 show the calculated appearance frequency of each value. Finally, the appearance-frequency values are used as features after normalization. When extracting these features from the test dataset, we use the appearance frequency calculated from the training dataset if the same values exist in the training dataset. If they do not, the value of the feature is set to 0.

For example, if given the training dataset on the left side in Fig. 4 and the values of an rDNS-e2LD, and FQDN are `ispX.example` and `www.example1.jp`, respectively, the feature values are determined as follows: $rDNS\text{-}e2LD = 0.429$, $FQDN = 0$, $rDNS\text{-}e2LD \text{ and } FQDN = 0$.

With the proposed method, we use three additional features. Categorical features (geoip) were vectorized by one-hot-encoding.

- **geoip:** The country of the client IP address determined by GeoIP [22] is used as a feature. Though an attacker might use many IP addresses to distribute sources of attacks, preparing IP addresses from various countries is generally costly. This feature was developed on the basis of the assumption that the geolocation of source IP addresses used in attacks tends to be a specific country.
- **dynamic_range:** Whether a client IP address is dynamically assigned by an ISP is used as a feature. On the basis of prior research [23]–[25], we apply sim-

Fig. 4 Procedure of feature calculation with proposed method

ple keyword matching (e.g., *dhcp*, *ppp*) to the rDNS lookup result of the client IP address. We consider this feature because it is a network-level attribute that cannot be changed by an attacker.

- **alexa_rank:** We use the list of Alexa Topsites [26] to measure the popularity of publisher websites. Specifically, we extract FQDNs from publisher URLs and match them with the list of Alexa's top 1 million sites. If a FQDN is contained in the list, the rank value is used as a feature. If it was not, the value is set to 1 million + 1. We assume that a website abused by automated ad fraud does not contain content that attracts normal users because such websites are costly to maintain, thus the rank value of such a websites tends to be low.

Conventional Features: Our method uses two burstiness-based features that are referred to in a prior study [11]. Though this simple metric can easily be changed by an attacker, it is still useful for detecting unsophisticated attacks.

- **impression_count:** The number of impressions from each client is used as a feature. We count the number of the impressions per client in the training dataset and normalize it. When calculating features of the test dataset, the value from the training dataset is used if ad requests from the same client exist in both training and test datasets, and the value is set to 0 if not.
- **ctr:** The click-through-rate (CTR), which is the rate of user's clicks per impression, is used as a feature. We calculate the CTR of each client in the training dataset, and the feature for the test dataset is calculated in the same manner as *impression_count*.

To calculate the above feature values, we have to identify each client and their ad requests. In this study, we identified each client by a unique combination of an IP address and user agent. The limitations of this approach are discussed later in Sect. 7.

4.3 Learning and Classification

The machine-learning model was constructed using a training dataset with the ground truth label. We used Random Forest [27] as our machine-learning algorithm because it is

accurate and can process a large amount of data in parallel. Random Forest is also superior to other machine-learning algorithms in terms of interpretability of trained models. We can obtain importance scores of each feature from a trained model, which represent how much a feature contributes to classification. The parameters of Random Forest were experimentally determined. This means we conducted a pilot experiment with a small dataset and determined the most promising parameters before evaluation. If there was a class imbalance in the training dataset, we sampled the dataset with proper weight to equalize the number of benign and malicious logs.

5. Evaluation

We evaluated the effectiveness of the proposed method. More precisely, we compared our features with conventional ones that are based on the burstiness of ad requests. We also evaluated the performance degradation over time by using long-term datasets. Finally, we empirically analyzed ad frauds in the wild by applying the proposed method to a large amount of dataset observed on an actual ad network over 24 hours. Our proposed method was implemented using the Python and scikit-learn [28] package. When extracting the OS version from user agent strings, we used ua-parser [29]. The experimental evaluation was conducted on an Ubuntu server with 16-core 2.8GHz Xeon CPUs and 128-GB RAM.

5.1 Dataset

Table 2 shows the details of datasets used in our evaluation. Dataset-A, which consists of whole ad-requests logs observed within the 24 hours, was used for evaluating the effectiveness of the proposed method and analyzing ad frauds in the wild. Datasets-B1 to B4 were used for evaluating performance degradation over time. Note that datasets-B1 to B4 are sampled datasets that were constructed during one specific hour and randomly sampled after data collection. The impression log indicates that an advertisement was displayed on a browser. The click log indicates that a user clicked an advertisement. These logs were collected on the actual ad network in Japan by an anti-ad-fraud ser-

Table 2 Breakdown of datasets used in evaluation

Name	# of impression logs	# of click logs	Date (duration)	Has ground truth label?
Dataset-A	13,814,768	37,970	29 Oct. 2017 (24H)	Partially (1% of all data)
Dataset-B1	208,645	708	6 June 2018 (1H)	Yes
Dataset-B2	242 411	893	13 June 2018 (1H)	Yes
Dataset-B3	253,083	978	20 June 2018 (1H)	Yes
Dataset-B4	323,074	954	27 June 2018 (1H)	Yes

vice provider [30]. The impression log contains a URL of a publisher's website displaying an advertisement (publisher URL), an IP address of an end user who sees the advertisement (client IP), and user-agent strings of the user's browser. The click log contains the X-Y coordinates of clicked position within the ad space and the timestamp of when a click occurred. The click log also contains impression IDs, thus we can determine which impression is tied to a specific click event. In the rest of the evaluation, we analyzed impression and click logs joined by impression ID.

In this evaluation, we manually established the ground truth. In supervised-learning approaches, the ground truth is essential. However, as mentioned in prior research [8], the ground truth about fraudulent and legitimate ad requests is difficult to establish. For example, to definitely distinguish whether a click is ad fraud, we must ask users whether they clicked the ad intentionally. This is impractical and almost impossible. This time, we used a small dataset that was manually inspected by operators of the log provider. The actual procedure of manual inspection and the detection algorithm are proprietary, but the official website of the log provider states that they integrate certain metrics (e.g., browser information, IP address, behavior of audience, and known attacker groups) to detect ad frauds [31].

5.2 Effectiveness of Proposed Method

We evaluated the effectiveness of the proposed method by using small datasets. First, we randomly selected 1% of logs from dataset-A consists of 138,136 records and manually annotated ground truth labels. Next, we constructed 12 pairs of training and test datasets with ground truth labels. In this evaluation, the parameters of time window Δt_1 and Δt_2 were set to 12 hours and 1 hour, respectively, and time point t was shifted by 1 hour from 12:00 to 23:00. In summary, the evaluation was conducted with 12 pairs of training and test datasets. For example, the data for the first training dataset were collected from 00:00 to 12:00 and corresponding test data were collected from 12:00 to 13:00. We used four metrics to quantify the effectiveness of the proposed method: precision, recall, receiver operating characteristic (ROC) curve, and area under the ROC curve (AUC). The scores of precision and recall are defined by the following equations:

$$\text{precision} = \frac{\text{Number of correctly detected fraudulent ad requests}}{\text{Number of all detected ad requests}}$$

$$\text{recall} = \frac{\text{Number of correctly detected fraudulent ad requests}}{\text{Number of all fraudulent ad requests in test dataset}}$$

In the context of ad-fraud detection, the higher the pre-

cision rate, the more revenue for publishers is saved. Also, the higher the recall rate, the more advertising funds for advertisers is saved. Ad networks are not directly affected by both precision and recall from the viewpoint of revenue, but the reputation of ad networks may be affected by detection performance. As stated in 2.2, we focused on ad frauds targeting advertisers' funds. Therefore, we focused more on the recall rate than the precision rate in this evaluation.

To evaluate the effectiveness of the machine-learning model trained using our features, we also trained three other models that only use conventional features: *impression_count*, *ctr*, and both *impression_count* and *ctr*. Some methods used attack properties other than burstiness of ad requests. However, because these methods require more ad-request properties, they can only be applied to datasets with rich information; thus, their applicability is limited. Because these methods require ad-request properties not contained in our datasets (e.g., an identifier of publisher, price of advertising units, etc.), they cannot be directly compared with our proposed method. Therefore, we compared our proposed method with the simplest burstiness-based method as a baseline. Comparison with state-of-the-art methods is for future work.

Figure 5 shows the boxplot of precision, recall, and AUC of each model. Though our model had slightly lower precision than the conventional ones, it had higher recall in all datasets. Average precision and recall were respectively 0.91 and 0.83 for our model and 0.97 and 0.73 for the best performing conventional model (only using *impression_count*). This indicates that our features contributed to improving the recall rate and reducing false negatives. The total number of reduced false negatives was about 650–1,000. In this evaluation, the number of ad-request logs contained in whole test dataset was 86,523, and the number of ad-request logs contained in dataset-A, which consists of whole ad-requests logs in a day, was 13,814,768. Therefore, the results indicate that our proposed method had about 100,000–160,000 fewer false negatives per day than the conventional methods when we applied it to the whole dataset-A. Figure 6 shows the ROC curve of each method. Note that each ROC curve was constructed by concatenated the results of each dataset. The proposed method had about 8.1–11.6% higher AUC than conventional ones.

Table 3 shows the top 20 features and average importance score of the model trained with the these features. We found that all the features based on appearance frequency were in the top 6. We also found that the importance of conventional features accounts for about 43% of total importance. This means that these simple features are still im-

Fig. 5 Boxplot of precision, recall, and AUC for each model

Fig. 6 ROC curve of classification results

Table 3 Feature importance scores of our features used to train machine-learning model (Top 20)

Feature name	Importance	Feature type
impression_count	0.27931	Conventional
rDNS-e2LD_freq	0.19556	Proposed
ctr	0.14630	Conventional
geoip_JP	0.12422	Proposed
rDNS-e2LD and FQDN	0.07677	Proposed
FQDN_freq	0.05661	Proposed
geoip_US	0.05029	Proposed
geoip_None	0.03566	Proposed
alexa_rank	0.01953	Proposed
is_dynamic_range	0.01221	Proposed
geoip_TW	0.00278	Proposed
geoip_CN	0.00014	Proposed
geoip_MY	0.00009	Proposed
geoip_TH	0.00008	Proposed
geoip_CA	0.00007	Proposed
geoip_KR	0.00006	Proposed
geoip_HK	0.00006	Proposed
geoip_PH	0.00005	Proposed
geoip_VN	0.00004	Proposed
geoip_SG	0.00004	Proposed

portant for detecting ad frauds.

Comparing the run-time performance of each method, there were almost no differences between proposed method and conventional ones. Specifically, the proposed method took about 5–10 additional seconds to extract features from the training dataset containing about 50,000 logs (equiva-

lent to 0.0001–0.0002 seconds/log), but this was sufficiently short and had a negligible effect.

5.3 Performance Degradation over Time

We applied the proposed method and the conventional burstiness-based one to long-term datasets to evaluate performance degradation over time. Dataset-B1, -B2, -B3, and -B4 shown in Table 2, which consist of ad-requests logs observed over multiple days within a month, were used in this evaluation. We conducted two experiments: first, we used ad-request logs from the first 30 minutes in dataset-B1 as a training dataset and those from the latter 30 minutes in datasets-B1 to -B4 as test datasets. This experimental setup was to evaluate non-retraining models. Specifically, the model trained from dataset-B1 was evaluated without retraining by using the test dataset constructed a few days later in the month. Second, we used ad-request logs from the first 30 minutes in datasets-B1 to -B4 as training datasets and those from the latter 30 minutes as test datasets. This was to evaluate retraining models. Specifically, a model was re-trained every week by using datasets constructed in the last 30 minute. Note that the test datasets used in both experiments were the same.

Figure 7 shows the precisions, recalls and AUCs for the test dataset from each day. Note that solid lines represent the results of retraining models and dashed ones represent the result of non-retraining models. We found that precision, recall and AUC of our proposed method was stable on each day. Also, there was almost no difference between the results of the non-retraining and retraining models. On the other hands, the detection performance of the conventional method was unstable, and that with non-retraining model decreased gradually over time. This indicates that our proposed method is more robust than the conventional one against performance degradation over time, and our newly designed features, which leverage the statistics of publishers and clients, are more stable than the burstiness of ad requests.

5.4 Contribution of Proposed Features

As mentioned in Sect. 3, the main difference between pro-

Fig. 7 Performance degradation of precision, recall and AUC for the test dataset from each day

posed method and existing ones is the fundamental property of attackers used to distinguish fraudulent activities from legitimate ones. From the evaluation in Sect. 5.2, we confirmed that proposed method had fewer false negatives comparing with the burstiness-based method. This indicate that our proposed features calculated from statistics of clients and publisher information contributed to distinguish fraudulent activities with low burstiness of ad requests. Also, we confirmed our proposed features are more stable than the burstiness-based features from the result of Sect. 5.3.

6. Large-Scale Measurement

We empirically analyzed ad fraud in the wild with the large-scaled dataset. Specifically, we classified the latter half of dataset-A by using the trained models presented in Sect. 5.2. Note that we applied our proposed method to test ad-requests logs without ground truth labels in this empirical analysis. The rest of this section describes the results of empirical analysis.

6.1 Basic Statistics

Table 4 shows the breakdown of detected impressions and clicks. The middle column shows the number of total impressions and clicks contained in the latter half of dataset-A, and the last column shows the number of detected ones and its percentage of total. We found that 588,274 of all 8,652,857 impressions (6.8%) were detected. Table 5 shows the mean, median, and standard deviation of the number of ad requests sent from each unique IP address and publisher URL. Note that we regarded IP addresses and publisher URLs as detected if there is at least one detected ad request containing these values. We confirmed that detected IP addresses and publisher URLs cause more ad requests than that of non-detected ones.

6.2 Analysis of Client IP Addresses

We investigated the ad requests sent from each rDNS-e2LD. We observed 2,641 rDNS-e2LDs. Figure 8 shows the top 50 rDNS-e2LDs sorted in the order of total number of detected ad requests from each rDNS-e2LD. The bars rep-

Table 4 Detected ad requests

Description	All	Detected (rate)
# of impressions	8,652,857	588,274 (6.8%)
# of clicks	22,213	1,224 (5.5%)

Table 5 Basic statistics of the number of ad requests sent from each unique IP address and publisher URL

Description	# of unique values	Mean	Median	S.D.
IP addresses (All)	2,333,456	3.71	2	131.01
IP addresses (Detected)	15,324	61.22	4	1614.36
URLs (All)	2,449,307	3.42	1	76.11
URLs (Detected)	191,358	18.44	2	265.60

Fig. 8 Detected impressions sent from each rDNS-e2LD (Top 50)

resent the total number of detected ad requests originating from client IP addresses with a common rDNS-e2LD, and lines represent the percentage of detected ad requests of all ad requests from client IP addresses with a common rDNS-e2LD (referred to as the fraud rate). Note that the y-axis of the bar chart is a logarithmic scale. We found that an enormous amount of detected ad requests were sent from specific rDNS-e2LDs. Specifically, about 45% of total detected ad requests originated from a single rDNS-e2LD. We also found that there are several rDNS-e2LDs with high fraud rates. The fraud rate on 7 of top 50 rDNS-e2LDs were over 80%. These results indicate that several group of IP addresses with common rDNS-e2LD tend to be abused by ad-fraud attacks. Table 6 shows the breakdown of detected ad requests sent from the top 20 rDNS-e2LDs. Note that actual values of rDNS-e2LDs were omitted. We manually inspected each rDNS-e2LD and found the organization that manages the IP addresses with the rDNS-e2LD and its purpose. In the category column, *cloud service* means that the

Table 6 Breakdown of detected ad requests sent from each rDNS-e2LD and rDNS-e2LD category (Top 20)

rDNS-e2LD	# of detected impressions (fraud rate)	# of unique IP addresses (fraud rate)	# of unique clients	rDNS-e2LD category
rDNS-e2LD.1	269,214 (99.9%)	339 (77.9%)	418	Cloud service
rDNS-e2LD.2	45,539 (85.0%)	129 (93.0%)	16,779	Proxy service
rDNS-e2LD.3	30,284 (3.8%)	162,843 (6.3%)	345,176	Mobile carrier
rDNS-e2LD.4	3,932 (87.8%)	65 (21.5%)	150	Cloud service
rDNS-e2LD.5	1,891 (99.1%)	3 (100.0%)	184	Cloud service
rDNS-e2LD.6	1,486 (0.4%)	119,286 (1.0%)	125,898	ISP
rDNS-e2LD.7	1,057 (99.1%)	1 (100.0%)	97	Cloud service
rDNS-e2LD.8	1,002 (0.1%)	321,989 (0.2%)	343,412	ISP
rDNS-e2LD.9	739 (0.9%)	5,939 (5.6%)	27,835	ISP
rDNS-e2LD.10	677 (0.1%)	273,584 (0.2%)	288,615	ISP
rDNS-e2LD.11	539 (0.1%)	210,626 (0.1%)	231,427	ISP
rDNS-e2LD.12	351 (31.1%)	6 (100.0%)	90	ISP
rDNS-e2LD.13	292 (0.2%)	42,945 (0.4%)	45,592	ISP
rDNS-e2LD.14	244 (0.5%)	10,551 (1.2%)	16,998	ISP
rDNS-e2LD.15	202 (44.0%)	180 (53.9%)	205	Cloud service
rDNS-e2LD.16	121 (2.5%)	609 (2.6%)	1,304	ISP
rDNS-e2LD.17	112 (0.2%)	20,300 (0.2%)	22,616	ISP
rDNS-e2LD.18	107 (<0.1%)	75,802 (0.1%)	338,795	Mobile carrier
rDNS-e2LD.19	94 (84.7%)	67 (95.5%)	92	ISP
rDNS-e2LD.20	85 (0.4%)	2,297 (2.0%)	7,080	ISP

Fig. 9 Detected impressions sent from each FQDN (Top 50)

IP addresses with this rDNS-e2LD are managed by a cloud and vps service providing virtual machines to end users. *Proxy service*, *Mobile carrier*, and *ISP* mean that the IP addresses with these rDNS-e2LDs are managed by a web proxy service, mobile network operator, and ISP, respectively. As the fraud rates of rDNS-e2LDs managed by cloud services are extremely high, we assume these cloud services tend to be abused by attackers. We also assume attackers frequently abuse the web proxy of rDNS-e2LD.2 to distribute their source IP addresses when launching attacks because the number of unique clients of rDNS-e2LD.2 is high.

6.3 Analysis of Publisher FQDN

We investigated ad requests sent from each FQDN of publisher URLs. We observed 113,860 FQDNs in this evaluation. Figure 9 shows the top 50 FQDNs sorted in the order of the total number of detected ad requests from the FQDN. The bars and lines represent the same values as those in Fig. 8. There were two FQDNs with especially high fraud rates. It is also noteworthy that certain percentage of ad frauds existed on the top 50 FQDNs. Specifically, fraud rate on 41 of top 50 FQDNs were over 10%. Table 7 shows the breakdown of the detected ad requests sent from the top 20 FQDNs. Note that actual values of FQDN were omitted. We manually inspected each FQDN to reveal who manages it and how it is used. The results are listed in the last col-

Table 7 Breakdown of detected impression sent from each FQDN and FQDN category (Top 20)

FQDN	# of detected impressions (fraud rate)	# of unique IP addresses (fraud rate)	# of unique clients	FQDN category
FQDN.1	84,111 (15.2%)	157,196 (0.2%)	160,702	Video-streaming site
FQDN.2	21,383 (9.5%)	97,459 (0.5%)	110,166	Blogging service
FQDN.3	14,168 (14.5%)	33,422 (0.2%)	36,071	Rewards site
FQDN.4	11,087 (3.2%)	129,560 (2.9%)	162,177	News site
FQDN.5	9,991 (20.2%)	17,239 (0.9%)	17,346	E-commerce site
FQDN.6	9,646 (14.0%)	20,584 (0.1%)	20,717	Video-streaming site
FQDN.7	9,392 (31.6%)	4,227 (0.5%)	4,294	Rewards site
FQDN.8	9,391 (23.4%)	16,058 (0.4%)	16,169	News site
FQDN.9	9,173 (11.5%)	35,169 (0.3%)	36,796	News site
FQDN.10	8,466 (4.2%)	73,825 (2.2%)	87,187	Forum site
FQDN.11	7,230 (99.3%)	258 (99.6%)	729	Hosting service
FQDN.12	6,633 (13.8%)	19,190 (0.2%)	19,342	News site
FQDN.13	6,139 (14.3%)	17,580 (0.1%)	17,682	Video-streaming site
FQDN.14	5,946 (10.6%)	17,170 (0.4%)	18,002	Forum site
FQDN.15	5,827 (19.4%)	5,022 (0.4%)	5,107	Rewards site
FQDN.16	4,738 (4.1%)	39,041 (0.7%)	39,842	Forum site
FQDN.17	4,411 (18.4%)	5,822 (0.5%)	5,877	Video-streaming site
FQDN.18	4,301 (10.5%)	21,965 (1.1%)	23,947	Forum site
FQDN.19	4,091 (3.2%)	38,610 (1.6%)	47,991	Forum site
FQDN.20	3,918 (99.8%)	3 (100.0%)	12	Unknown

umn. We found the fraud rates of FQDN.11 and FQDN.20 were much higher than others. FQDN.11 is managed by a hosting service usually used to deliver web content. Also, by manually inspecting the publisher URLs with FQDN.11 and FQDN.20, we observed many URLs that host web content only containing an ad-tag. This means only an advertisement is displayed in the browser frame when these URLs are accessed. We also found that FQDN.3, FQDN.7 and FQDN.15 were used in rewards websites. On rewards websites, users can obtain points, which can be exchanged for actual money in many cases, by doing tasks such as registering an account of a web service, playing simple browser games, and answering a survey. This type of website usually contains many ads on the task pages. We assumed that detection of ad requests from these rewards websites was caused by end users who automate the given tasks to fraudulently earn points. Although we cannot regard the rewards websites themselves as fraudulent web services as long as they obtain user consent and the ads contained in their task pages are legitimate, we assume their service designs might cause ad frauds. In addition to the above categories, we found several types of websites (e.g., the news websites, blogging services, and forum websites) were abused by ad frauds.

6.4 Analysis of Client Environment

We investigated ad requests sent from each OS version of clients. We observed 56 types of OS versions. Table 8 shows the number of detected ad requests and fraud rates of the top 20 OS versions. In our datasets, we observed that a large amount of ad requests from Windows XP was detected. Specifically, 83% of all detected ad requests was sent from Windows XP. We assumed that these ad frauds from Windows XP were caused by unsophisticated bots or automation tools that request ads with a pre-defined user-agent value. Note that our method did not use the infor-

Table 8 Detected ad requests sent from each OS version (Top 20)

OS version	# of detected impressions (fraud rate)	# of unique IP addresses (fraud rate)	# of unique clients
WindowsXP	493,233 (95.1%)	8,171 (0.9%)	8,212
Android7	30,948 (2.4%)	423,298 (0.7%)	543,686
Android5	16,770 (2.7%)	226,423 (1.8%)	269,125
Android4	14,650 (2.4%)	225,483 (1.7%)	280,049
Android6	12,169 (1.4%)	290,507 (0.3%)	362,672
Windows7	4,575 (0.4%)	345,008 (0.4%)	356,542
Windows10	4,242 (0.2%)	579,575 (0.3%)	602,586
iOS9	2,884 (4.1%)	26,943 (3.0%)	28,580
iOS10	1,836 (0.3%)	200,804 (0.5%)	217,204
iOS8	1,083 (4.5%)	9,249 (1.0%)	9,715
Windows8.1	941 (0.3%)	109,177 (0.2%)	111,147
MacOSX10	686 (0.7%)	32,804 (0.2%)	33,440
SymbianOS9	568 (93.1%)	31 (96.8%)	107
iOS11	533 (0.1%)	176,873 (0.2%)	185,747
Android8	526 (5.3%)	4,209 (0.7%)	4,404
WindowsPhone10	414 (51.6%)	178 (11.2%)	186
iOS5	379 (22.1%)	573 (5.2%)	584
Windows8	318 (2.4%)	4,854 (1.0%)	4,910
BlackBerryOS7	223 (94.1%)	24 (83.3%)	60
ChromeOS9765	217 (50.2%)	169 (62.7%)	169

mation about OS version at all, but it correctly captured the noticeable features of the client environment. It is also noteworthy that the amount of ad frauds from Android devices is also high.

7. Discussion

In this section, we discuss the limitations, future work, and ethics and privacy.

7.1 Limitations

Biased dataset: The datasets we analyzed were biased in terms of the observation point of ad requests and the period of observation. Table 9 shows the number of ad requests sent from the top 5 country. We found 95.3% of ad requests originated from Japan. This means the results from this evaluation represent the situation of ad fraud in Japan and might differ from the results for ad fraud observed in other countries. Also, our datasets were constructed within a specific ad network, though there are many other ad networks in the entire ad ecosystem. Our proposed method is based on the tendencies of ad requests within the studied ad network, thus the evaluation results might differ if we apply our method to datasets from other ad networks. Because our datasets were constructed within a specific period (24 hours in one day and 1 hour over several days), they might have lacked features that can be extracted from longer-term datasets. Analyzing a longer-term dataset is for future work.

Granularity of client identification: In this study, we identified each client by a unique combination of IP address and user agent. This approach results in overestimation when extracting features due to the existence of a Network Address Translation (NAT) and HTTP proxy. Therefore, we have to introduce a more precise client-identification

Table 9 Distribution of countries (Dataset-A)

Country	# of ad requests (rate)
Japan	13,165,257 (95.3%)
United States	459,331 (3.3%)
Taiwan	91,049 (0.7%)
Unknown	76,210 (0.6%)
Korea	3,090 (<0.1%)

method, such as HTTP cookies and browser fingerprints. However, an attacker may avoid even such massive tracking. Considering the above, fine-grained client identification is for future work.

7.2 Ethics and Privacy

Our study incorporated considerations regarding research ethics and privacy of end users (clients), publishers, and advertisers. Our dataset did not include advertiser information (the names of advertisers and advertisement content). We anonymized client IP addresses in our dataset after retrieving information corresponding to the client environment of client IP addresses, as explained in Sect. 4.2. We statistically conducted an analysis and did not infer or track users' attributions. All specific IP addresses and domain names were suppressed in this paper. Although most of the analysis was conducted offline, in some cases, we accessed specific websites. When accessing, we manually generated a limited number of requests in order not to violate the acceptable use policy (AUP) for each entity involved in an advertisement ecosystem or increase the load of the servers.

8. Conclusion

We proposed a method of detecting fraudulent ad requests observed within an ad network. Our proposed method leverages novel features designed on the basis of the assumption that an attacker cannot know of the statistics of ad requests from legitimate users such as the tendencies of client environments and publisher websites. By evaluating the effectiveness of our proposed method, we found that our newly designed features contributed to improving the recall rate by 10% more than the conventional features based on the burstiness of ad requests. We also confirmed that our proposed method was more robust against performance degradation over time compared with a conventional burstiness-based method. Finally, we conducted an empirical analysis of ad frauds by applying the proposed method to a large amount of ad-request logs collected within an actual ad network. We found that 6.8% of 8 million ad requests were detected by the proposed method and observed some characteristics of fraudulent ad requests. For instance, a large amount of fraudulent ad requests was sent from IP addresses assigned to cloud servers, and the client environments extracted from the fraudulent ad requests were heavily biased toward a specific OS version. We believe these findings reflect properties of current ad fraud in the wild and can be utilized for improving ad fraud detection techniques.

References

- [1] F. Kanei, D. Chiba, K. Hato, and M. Akiyama, "Precise and robust detection of advertising fraud," 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), pp.776–785, IEEE, 2019.
- [2] "The Interactive Advertising Bureau | IAB internet advertising revenue report." <https://www.iab.com/wp-content/uploads/2019/05/Full-Year-2018-IAB-Internet-Advertising-Revenue-Report.pdf>
- [3] "What Is An Untrustworthy Supply Chain Costing The U.S. Digital Advertising Industry?." https://www.iab.com/wp-content/uploads/2015/11/IAB_EY_Report.pdf
- [4] Y. Chen, P. Kintis, M. Antonakakis, Y. Nadji, D. Dagon, W. Lee, and M. Farrell, "Financial lower bounds of online advertising abuse," Proc. 13th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, DIMVA '16, pp.231–254, Springer, 2016.
- [5] Y. Chen, Y. Nadji, R. Romero-Gómez, M. Antonakakis, and D. Dagon, "Measuring network reputation in the ad-bidding process," Proc. 14th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, DIMVA '17, pp.388–409, Springer, 2017.
- [6] J. Crussell, R. Stevens, and H. Chen, "Madfraud: Investigating ad fraud in android applications," Proc. 12th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys '14, pp.123–134, ACM, 2014.
- [7] X. Xing, W. Meng, B. Lee, U. Weinsberg, A. Sheth, R. Perdisci, and W. Lee, "Understanding malvertising through ad-injecting browser extensions," Proc. 24th International Conference on World Wide Web, WWW '15, pp.1286–1295, International World Wide Web Conferences Steering Committee, 2015.
- [8] V. Dave, S. Guha, and Y. Zhang, "Measuring and fingerprinting click-spam in ad networks," Proc. ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication, SIGCOMM '12, pp.175–186, ACM, 2012.
- [9] V. Dave, S. Guha, and Y. Zhang, "ViceROI: Catching click-spam in search ad networks," Proc. 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS '13, pp.765–776, ACM, 2013.
- [10] T. Tian, J. Zhu, F. Xia, X. Zhuang, and T. Zhang, "Crowd fraud detection in internet advertising," Proc. 24th International Conference on World Wide Web, WWW '15, pp.1100–1110, International World Wide Web Conferences Steering Committee, 2015.
- [11] B. Stone-Gross, R. Stevens, A. Zarras, R. Kemmerer, C. Kruegel, and G. Vigna, "Understanding fraudulent activities in online ad exchanges," Proc. 2011 ACM SIGCOMM Conference on Internet Measurement Conference, IMC '11, pp.279–294, ACM, 2011.
- [12] N. Daswani and M. Stoppelman, "The anatomy of clickbot.a," Proc. First Conference on First Workshop on Hot Topics in Understanding Botnets, HotBots'07, pp.11–11, USENIX Association, 2007.
- [13] P. Pearce, V. Dave, C. Grier, K. Levchenko, S. Guha, D. McCoy, V. Paxson, S. Savage, and G.M. Voelker, "Characterizing large-scale click fraud in zeroaccess," Proc. 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS '14, pp.141–152, ACM, 2014.
- [14] B. Miller, P. Pearce, C. Grier, C. Kreibich, and V. Paxson, "What's clicking what? Techniques and innovations of today's clickbots," Proc. 8th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, DIMVA'11, pp.164–183, Springer, 2011.
- [15] N. Jagpal, E. Dingle, J. Gravel, P. Mavrommatis, N. Provos, M. Rajab, and K. Thomas, "Trends and lessons from three years fighting malicious extensions," Proc. 24th USENIX Conference on Security Symposium, SEC'15, pp.579–593, USENIX Association, 2015.
- [16] K. Thomas, E. Bursztin, C. Grier, G. Ho, N. Jagpal, A. Kapravelos, D. McCoy, A. Nappa, V. Paxson, P. Pearce, N. Provos, and M. Rajab, "Ad injection at scale: Assessing deceptive advertisement modifications," Proc. 2015 IEEE Symposium on Security and Privacy, SP '15, pp.151–167, IEEE, 2015.
- [17] A. Metwally, D. Agrawal, and A.E. Abbadi, "Duplicate detection in click streams," Proc. 14th International Conference on World Wide Web, WWW '05, pp.12–21, ACM, 2005.
- [18] L. Zhang and Y. Guan, "Detecting click fraud in pay-per-click streams of online advertising networks," Proc. 28th International Conference on Distributed Computing Systems, ICDCS '08, pp.77–84, IEEE, 2008.
- [19] A. Metwally, D. Agrawal, and A.E. Abbadi, "Detectives: Detecting coalition hit inflation attacks in advertising networks streams," Proc. 16th International Conference on World Wide Web, WWW '07, pp.241–250, ACM, 2007.
- [20] F. Yu, Y. Xie, and Q. Ke, "SBotMiner: Large scale search bot detection," Proc. Third ACM International Conference on Web Search and Data Mining, WSDM '10, pp.421–430, ACM, 2010.
- [21] "Mozilla Foundation | Public Suffix List." <https://publicsuffix.org/>
- [22] "MaxMind | GeoIP2 Databases." <https://www.maxmind.com/en/geoip2-databases>
- [23] Y. Xie, F. Yu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber, "How dynamic are IP addresses?," Proc. 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM '07, pp.301–312, ACM, 2007.
- [24] Y. Jin, E. Sharafuddin, and Z. Zhang, "Identifying dynamic IP address blocks serendipitously through background scanning traffic," Proc. 2007 ACM CoNEXT Conference, CoNEXT '07, pp.4:1–4:12, ACM, 2007.
- [25] X. Cai and J. Heidemann, "Understanding block-level address usage in the visible internet," Proc. ACM SIGCOMM 2010 Conference, SIGCOMM '10, pp.99–110, ACM, 2010.
- [26] "Alexa Top Sites." <https://www.alexa.com/topsites>
- [27] L. Breiman, "Random forests," Machine learning, vol.45, no.1, pp.5–32, 2001.
- [28] "scikit-learn." <http://scikit-learn.org/>
- [29] "ua-parser/uap-python." <https://github.com/ua-parser/uap-python>
- [30] "Momentum Inc." <http://www.m0mentum.co.jp/>
- [31] "Black Heron | Momentum Inc." <http://www.m0mentum.co.jp/en/service/blackheron.html>

Fumihiro Kanei received his M.E. degree in computer science from Yokohama National University in 2015. Since joining Nippon Telegraph and Telephone Corporation (NTT) in 2015, he has been engaged in research and development on cyber security, especially mobile security, web security, and program analysis. He is now a researcher in the Cyber Security Project of NTT Secure Platform Laboratories.

Daiki Chiba is currently a researcher at NTT Secure Platform Laboratories, Tokyo, Japan. He received his B.E., M.E., and Ph.D. degrees in computer science from Waseda University in 2011, 2013, and 2017. Since joining Nippon Telegraph and Telephone Corporation (NTT) in 2013, he has been engaged in research on cyber security through data analysis. He won the Research Award from the IEICE Technical Committee on Information and Communication System Security in 2016, 2018, and 2019 and

the Best Paper Award from the IEICE Communications Society in 2017. He is a member of IEEE and IEICE.

Kunio Hato received his B.E. and M.E. degrees in information processing from Tokyo Institute of Technology in 1997 and 1999, respectively. He joined the Nippon Telegraph and Telephone Corporation (NTT) in 1999, where he was previously engaged in research and development of IP VPNs, wide area Ethernet. He is now a Senior Research Engineer, Supervisor, in Cyber Security Project of NTT Secure Platform Laboratories. He was with the Network Services of NTT communications from 2014 to

2017. He is a member of IEICE.

Katsunari Yoshioka is an Associate Professor at Yokohama National University since 2011. His research interests cover wide area of system security and network security including malware analysis and IoT security. He received the commendation for science and technology by the minister of MEXT, Japan in 2009, the award for contribution to Industry-Academia-Government Collaboration by the minister of MIC, Japan in 2016, and the Culture of Information Security Award in 2017.

Tsutomu Matsumoto is a professor of Faculty of Environment and Information Sciences, Yokohama National University and directing the Research Unit for Information and Physical Security at the Institute of Advanced Sciences. He is also the Director of Cyber Physical Security Research Center (CPSEC) at National Institute of Advanced Industrial Science and Technology (AIST). He received Doctor of Engineering from the University of Tokyo in 1986. Starting from Cryptography in the early 80's, he has

opened up the field of security measuring for logical and physical security mechanisms. Currently he is interested in research and education of Embedded Security Systems such as IoT Devices, Cryptographic Hardware, In-vehicle Networks, Instrumentation and Control Security, Tamper Resistance, Biometrics, Artifact-metrics, and Countermeasure against Cyber-Physical Attacks. He is serving as the chair of the Japanese National Body for ISO/TC68 (Financial Services) and the Cryptography Research and Evaluation Committees (CRYPTREC) and as an associate member of the Science Council of Japan (SCJ). He was a director of the International Association for Cryptologic Research (IACR) and the chair of the IEICE Technical Committees on Information Security, Biometrics, and Hardware Security. He received the IEICE Achievement Award, the DoCoMo Mobile Science Award, the Culture of Information Security Award, the MEXT Prize for Science and Technology, and the Fuji Sankei Business Eye Award.

Mitsuaki Akiyama received his M.E. and Ph.D. degrees in information science from Nara Institute of Science and Technology, Japan in 2007 and 2013. Since joining Nippon Telegraph and Telephone Corporation (NTT) in 2007, he has been engaged in research and development on cybersecurity. He is currently a Senior Distinguished Researcher with the Cyber Security Project of NTT Secure Platform Laboratories. His research interests include cybersecurity measurement, offensive security, and usable

security and privacy.