Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Android Malware Detection Based on Composition Ratio of Permission Pairs

HIROYA KATO[1], (Graduate Student Member, IEEE), TAKAHIRO SASAKI[1], and IWAO SASASE.[1], (Senior Member, IEEE)

[1]Department of Information and Computer Science, Faculty of Science and Technology, Keio University 3-14-1 Hiyoshi, Kohoku, Yokohama, Kanagawa 223-8522, Japan

Corresponding author: Hiroya Kato (e-mail: kato@sasase.ics.keio.ac.jp).

ABSTRACT Detecting Android malware is imperative. Among various detection schemes, permission pair based ones are promising for practical detection. However, conventional schemes cannot simultaneously meet requirements for practical use in terms of efficiency, intelligibility, and stability of detection performance. Although the latest scheme relies on differences of frequent pairs between benign apps and malware, it cannot meet the stability. This is because recent malware tends to require unnecessary permissions to imitate benign apps, which makes using the frequencies ineffective. To meet all the requirements, in this paper, we propose Android malware detection based on a Composition Ratio (CR) of permission pairs. We define the CR as a ratio of a permission pair to all pairs in an app. We focus on the fact that the CR tends to be small in malware because of unnecessary permissions. To obtain features without using the frequencies, we construct databases about the CR. For each app, we calculate similarity scores based on the databases. Finally, eight scores are fed into machine learning (ML) based classifiers as features. By doing this, stable performance can be achieved. Since our features are just eight-dimensional, the proposed scheme takes less training time and is compatible with other ML based schemes. Furthermore, our features can quantitatively offer clear information that helps human to understand detection results. Our scheme is suitable for practical use because all the requirements can be met. By using real datasets, our results show that our scheme can detect malware with up to 97.3% accuracy. Besides, compared with an existing scheme, our scheme can reduce the feature dimensions by about 99% with maintaining comparable accuracy on recent datasets.

INDEX TERMS Android malware detection, Permission pairs, Composition ratio, Practical detection

I. INTRODUCTION

Android is the most popular smartphone platform occupying 85% of market share in the world [1]. Smartphone is an essential tool for many people, and Android apps can provide various services such as online banking and games. Because of its popularity, smartphones running on Android system have become the main target of attackers [2]. In the third quarter of 2019, approximately 365,000 new Android malware samples have been found [3]. Android malware is rapidly growing in an attempt to elaborately pretend benign apps. Thus, using Android apps entails the risk of installing malware, which results in the urgency of devising practical detection schemes.

To detect sophisticated malware on the basis of various aspects, many detection methods have been proposed. Existing detection schemes are roughly divided into three categories, namely “network traffic based schemes” [4], [5] and “inner interaction based ones” [6]–[9], “permission based ones” [10]–[14]. In network traffic based schemes [4], [5], network traffic is used to create features for detecting malware. These methods are helpful because most attackers are inclined to use network for achieving malicious purposes. However, network based methods are not applicable to malware that conducts attacks without networks. Besides, they inevitably require running apps to collect features, which results in a certain runtime overhead. On the other hand, in inner interaction based schemes [6]–[9], features are extracted from various factors such as Application Programming Interface (API) calls and Inter-Component Communication (ICC) patterns. These methods are instrumental in capturing detailed features of apps because there exists difference of the usage of the above factors between benign apps and malware. However, these types of schemes also suffer from runtime overhead to extract API calls and ICC patterns even if static analysis is adopted. In order to realize lightweight and efficient detection, permission based schemes [10]–[14] are useful because permissions can be statically extracted from apps. Thus, permission based schemes are promising for practical detection. Among various permission based schemes, "permission pairs" based schemes [12]–[14] are more effective and informative than a single permission based ones [10], [11]. In other words, using permission pairs improves detection performance and helps users and security engineers to understand detection results in the practical situations because the permission pairs can provide more detailed information about how permissions are used together. These facts motivate us to pay attention to permission pairs based schemes [12]–[14] as attractive ones. Although various permission pairs based schemes [12]–[14] have been proposed, none of them are able to simultaneously satisfy three critical requirements for practical use, which hinders the conventional schemes from being used practically. To be specific, it is desirable that the following three requirements are satisfied.

1) Efficiency: Practical detection systems need to realize accurate detection by using low-dimensional feature vectors and compact data, which achieves lightweight detection. Compact features make systems compatible with other machine learning (ML) based schemes, which enables hybrid detection with feature fusion.
2) Intelligibility: A practical scheme must offer clear information that helps human to understand detection results for justifying the results.
3) Stability: A detection scheme must be able to detect both old and recent malware samples.

These requirements are very important to deploy detection schemes in practical situations. However, none of the conventional permission pairs based schemes are able to simultaneously meet the three requirements. In particular, we found that even the latest scheme [14] cannot fulfill the stability because of artifices by recent malware whereas other requirements are met. This is because that scheme relies on the differences of frequencies of permission pairs between benign apps and malware samples. Note that the frequency of a pair is normalized in a range of 0 to 1 in that scheme. For example, when a pair appears in half of all the apps in a dataset, the frequency of the pair is expressed as 0.5. Recent malware tends to require unnecessary permissions as well to imitate benign apps. As a result, since more pairs appear in both benign apps and malware, the frequency based detection gets ineffective. Thus, this work aims to devise a practical scheme that meets the above requirements by using a new tendency.

In order to fulfill the three requirements, in this paper, we propose Android malware detection based on a Composition Ratio (CR) of permission pairs. The CR is defined as a ratio of a permission pair to all pairs in an app. For example, suppose that an app requires five permissions in its manifest file. In this case, since the number of permission pairs is 10, the CR of pairs in the app is $0.1 = \frac{1}{10}$. We focus on the fact that the CR tends to be small in malware compared to benign apps. The reason why there exists the difference of the CR is that unnecessary permissions tend to be contained in manifest files of malware samples to imitate benign apps. To obtain features without using the frequencies, we construct databases regarding the CR in training datasets. For each app, we calculate eight similarity scores on the basis of the databases. By doing this, even if recent malware requires unnecessary permissions, the helpful features of them can be obtained from a new aspect. Thus, the proposed scheme can satisfy "Stability". Finally, our scores are used as features in ML for detection. Since the proposed scheme utilizes just eight-dimensional features, it can realize reducing computational cost of training and compatibility with other features, which results in satisfaction of "Efficiency". Besides, our scores can quantitatively offer clear information that what types of permission pairs dictate detection results. Thus, since numerical evidence is provided for human, the proposed scheme also meets "Intelligibility". Since all the requirements can be met, our scheme is more suitable for practical use.

The contributions of this paper are as follows:

1) We discovered the fact that permissions pairs in benign apps also tend to appear frequently in recent malware samples. This fact causes degradation in detection performance of the latest permission pairs based scheme.
2) We propose a more practical malware detection scheme using the CR of permission pairs. To the best of our knowledge, this work first proposes using the CR for malware detection.
3) The proposed scheme enables to not only realize stable detection but also offer numerical evidence to human.
4) Our scheme is compatible with other ML based schemes because our features are just eight-dimensional ones, which enables hybrid detection.

The rest of this paper is constructed as follows. Related work is introduced in Section II. Motivation of this work is described in Section III. The shortcoming of conventional schemes are explained in Section IV. The proposed scheme is presented in Section V. Evaluation results and discussion are shown in Section VI. Limitations of our scheme are described in Section VI. Finally, the conclusion of this paper and future work are presented in Section VIII.

II. RELATED WORK

Android malware detection schemes have been proposed on the basis of various aspects in order to detect sophisticated malware. In this section, we introduce representative schemes. Although Android malware detection schemes are often classified into static, dynamic, and hybrid detection, we classify existing methods on the basis of categories of features. This is because we consider that a type of feature is the primary factor in dictating detection performance which is the most important requirement in malware detection. Android malware detection methods are roughly divided into three categories, namely "network traffic based methods" [4],

[5] and "inner interaction based ones" [6]–[9], "permission based ones" [10]–[14].

A. NETWORK TRAFFIC BASED SCHEME

In network traffic based schemes [4], [5], network traffic is used to create features to detect malware. In order to obtain solid evidence of malware, Wang *et al.* [4] propose a method that pays attention to the occurrence of words regarding sensitive information in HTTP header of traffic data. Malware uses HTTP-POST/GET schemes for sending sensitive information. Therefore, semantic text features can be extracted from HTTP packets. The semantic text features mean the words such as "latitude", "longitude", and "imei", which is the unique identifier of a phone. That scheme can effectively detect malware by using dynamic evidence because most attackers are inclined to use network for achieving malicious purposes. However, recent malware cannot be efficiently detected by that scheme because they tend to encrypt malicious payloads.

To address encrypted traffic data, Garg *et al.* [5] propose a scheme that leverages network traffic patterns in benign apps and malware. The main idea of that scheme is that there exists the difference of network traffic patterns between benign apps and malware. That scheme is helpful because traffic patterns can be obtained even if malicious traffic is encrypted. However, the method is not applicable to malware that conducts attacks without network. Thus, detection schemes that are able to deal with various types of malware samples are also needed.

B. INNER INTERACTION BASED SCHEME

In inner interaction based schemes, features are extracted from factors such as API calls and ICC used in an app.

As an element that is useful in obtaining detailed features regardless of types of malware, API is very promising. Aafer *et al.* [6] propose a scheme using top-169 API calls which appear more frequently in malware than in benign apps to differentiate patterns of them. As a similar scheme, Deshotels *et al.* [7] propose a scheme focusing on the fact that malware abuses sensitive API call. In that scheme, malware samples are classified in accordance with malicious patterns predefined in advance. However, it is difficult for these schemes to detect the malware which carries out an attack by colluding with another malware because none of sensitive API calls are used by the main malware.

To deal with sophisticated attacks such as a collusion attack, Xu *et al.* [8] propose a scheme called ICCDetector focusing on the difference of ICC patterns between benign apps and malware. ICC is inner communication among Android system and apps. As a rule, ICC is mainly used for internal communication in a benign app. Nevertheless, malware tends to communicate with other apps via ICC to carry out malicious actions. Thus, ICCDetector can effectively detect attacks including the collusion attack. However, it is difficult for ICCDetector to deal with malicious behaviors that are conducted through obfuscation techniques called "reflection" because sensitive APIs are called via dynamic code constructs.

To cope with such clever techniques, Cai *et al.* [9] propose a scheme called DroidCat, which uses a diverse set of dynamic features based on ICC Intents. DroidCat adopts a purely dynamic approach that resolves reflective calls at runtime. Hence, that scheme is fully resilient against reflection. However, DroidCat inevitably requires running apps to collect dynamic features, which results in a certain runtime overhead. Considering a huge amount of new malware is rapidly increasing, lightweight and efficient methods are desired.

C. PERMISSION BASED SCHEME

Permission based schemes are useful in realizing lightweight and efficient detection methods. Sanz *et al.* [10] propose a detection method using permissions extracted from apps. They noticed there exist several difference of required permissions in benign apps and malware. However, since that work just reveals that permissions are useful for detection, significant permissions were not inspected at all.

To efficiently detect malware with fewer permissions, Li *et al.* [11] propose a method that only utilizes significant permissions extracted from apps. The idea of that method is that malware tends to require the common permissions that enable to conduct high risk operations such as accessing device information and personal information of users. Instead of extracting and analyzing all permissions, that scheme conducts three levels of pruning by mining the permission data to identify the most significant permissions for detection. That scheme can maintain high detection accuracy with just 22 permissions. However, in [10], it is reported that the top five permissions in both the categories are exactly the same. Thus, it is important to consider how such common permissions are used with other permissions in benign apps and malware.

Liang *et al.* [12] propose a rule based detection scheme by using permission combination. The permission combination is useful to express the relationship between a permission and another. They develop a tool that automatically generates rule sets based on frequencies of permission combinations to detect malware. That scheme can accurately detect malware samples when using a permission group of six. However, utilizing permissions in a group of more than two can lead to a large number of patterns, and its expression will be complex. Thus, detection with a group of more than two is not favorable for practical application because memory space consumption and computational cost will get high.

Similarly, Liu *et al.* [13] propose a detection method using permission pairs. The pairs are more useful than the combination of six permissions in terms of computational cost. That scheme converts the requested permission pairs into a boolean vector because such a vector can represent information about pairs. The boolean vector is fed into ML to detect malware. Permission pairs are constructed only from top 40 permissions that are frequently used in benign apps and malware samples because utilizing all permission pairs considerably increase computational cost and memory consumption. However, the boolean vector inevitably gets huge as considered permissions are increased in order to achieve high detection performance. In other words, there exist a trade-off between detection accuracy and computational cost. Furthermore, in light of a practical use, the boolean vector is not sufficient to provide users with convincing information of analysis. It is not easy for human to understand the meaning of a boolean vector whereas understandable information is important to justify detection results. Thus, how to represent permission pairs for practical detection is one of the important goals.

To efficiently represent the permission pairs, Arora *et al.* [14] propose a detection scheme called PermPair, which is based on frequencies of permission pairs in apps. That scheme leverages the fact that frequencies of permission pairs appearing in malware samples are different from the ones in benign apps. PermPair constructs databases on the basis of frequencies of permission pairs in prepared datasets of benign apps and malware. The frequency of a pair is normalized in a range of 0 to 1. In other words, when a pair appears in half of all the apps in a dataset, the frequency of the pair is expressed as 0.5. By using the databases, PermPair calculates two similarity scores which represent how similar a characteristic of an app is to benign apps and malware samples in databases. When a pair exists in a database, its normalized frequency is simply added to the similarity score. PermPair can convert permission pairs based information into just two scores, which results in providing clear information for human. By comparing the two scores, PermPair can judge whether an app is malware or not and realize lightweight detection without ML.

III. MOTIVATION OF THIS WORK

Although various schemes have been proposed, permission based schemes play an important role. This is because permissions are essential factors to develop apps, and malware inevitably requires them to carry out malicious actions. Furthermore, permission based schemes [10]–[14] are lightweight compared with other detection schemes using features such as network traffic [4], [5], API calls [6], [7], and ICC [8], [9]. In light of the predicament that a huge amount of malware is rapidly increasing, it is desirable that permission based detection schemes are practically used as a lightweight filter. In particular, permission based detection is promising for a filter in the first step to identify malware. Therefore, detecting more malware samples accurately by using permissions is very important to quickly deal with a huge amount of malware. After permission based detection, other schemes should be applied to tested apps if needed.

Among permission based schemes, utilizing permission pairs are more informative than a single permission. In other words, it is easier for users and security engineers to understand detection results because the permission pairs can represent how permissions are used together. Furthermore, in [14], it is reported that permission pairs based scheme is more effective than single permission based ones as for detection accuracy. These facts motivate us to pay attention to permission pairs based schemes [12]–[14] as conventional ones. However, none of conventional schemes cannot simultaneously satisfy the three requirements, namely "Efficiency", "Intelligibility", and "Stability" explained in Section I. In the subsequent section, we elaborate on the shortcomings. Thus, this work aims to devise a more practical scheme that meets the above requirements.

IV. SHORTCOMING OF CONVENTIONAL SCHEMES

To devise more practical schemes, detection systems must realize "Efficiency", "Intelligibility", and "Stability". These requirements are very important to deploy detection schemes in practical situations. Table 1 shows comparison of conventional permission pairs based schemes. As shown in Table 1, there exist strong points and weak points in every scheme [12]–[14]. Rule Set Based Scheme (RSBS) [12] and Boolean Vector Based Scheme (BVBS) [13] only satisfy "Intelligibility" and "Stability", respectively. On the other hand, although PermPair [14] can meet "Efficiency" and "Intelligibility" by using the two simple scores, it cannot meet "Stability". In the following subsections, we describe the above requirements in detail.

1) Efficiency

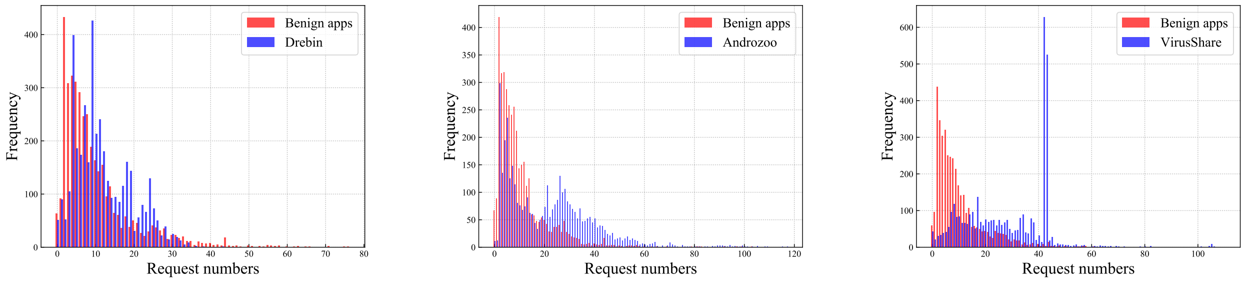
With regard to "Efficiency", RSBS [12] and BVBS [13] have weak points. In [12], it is reported that RSBS cannot realize accurate classification with permission pairs unless it uses combinations of six permissions. Therefore, RSBS is not efficient because it needs to process numerous combinations of six permissions to create an effective rule set.

In BVBS [13], boolean vectors tend to be large although such vectors can simply represent information about permission pairs. For example, when 40 permissions are used for detection, ${}_{40}C_2 = 780$ pairs are created, which means 780-dimensional sparse vectors. Hence, such feature vectors are not efficient in terms of converting information about the pairs into features. These schemes need huge memory to store boolean vectors and combinations based rule set because the dimensions of the boolean vector and the number of rules inevitably get huge.

Furthermore, most existing detection schemes adopt ML [4]–[6], [8]–[11], [13], because it can yield high detection accuracy. In every ML based scheme, various features have been defined. The various features can be used together for enhancing detection performance, which results in inevitably increasing required space, training time, and complexity of model. In light of the fact that huge training apps and regularly updating training data are required for high detection performance, it is desirable that dimensions of features are low to reduce the computational cost and complexity. This is why converting information about permission pairs into low-dimensional features is important. Since low-dimensional features can also provide compatibility with other schemes,

TABLE 1. Comparison of conventional permission pairs based schemes.

Reference	Notation of Schemes	Mechanism	Requirement		
			Efficiency	**Intelligibility**	**Stability**
Liang *et al.*(2014) [12]	RSBS	Frequency based rule set	Low	High	Low
Liu *et al.*(2014) [13]	BVBS	boolean vectors	Low	Low	High
Arora *et al.*(2019) [14]	PermPair	Frequency based score	High	High	Low



(a) Drebin (2010–2012) (b) Androzoo (2017–2020) (c) VirusShare (2018–2020)

FIGURE 1. Distribution of the number of permissions required by benign apps and malware samples.

for the purpose of realizing hybrid detection systems in the real world, this requirement must be satisfied.

2) Intelligibility

As for "Intelligibility", BVBS [13] has a drawback. Boolean vectors are not informative enough to describe the reason why an app is judged as malware, and vice versa. This problem was also suggested in the other existing work [15]. On the other hand, RSBS [12] and PermPair [14] can offer understandable information about detection results because rule set and scores can be utilized for interpretation. In the practical use, clear information should be provided for users to understand detection results without adequate expertise. Besides, security engineers need to understand detection results so that detailed analysis can be carried out. Since the intelligibility can bring reliability of detection, this requirement must be met.

3) Stability

In malware detection, "Stability" is the most important requirement. In terms of "Stability", the RSBS [12] and PermPair [14] have weak points. In [12], it is reported that frequency based rule set created from permission pairs cannot accurately distinguish benign apps from malware. According to results in [12], more than 55% of benign apps are misjudged as malware whereas 99% of malware samples are detected. On the other hand, we found that PermPair [14] is also incapable to detect recent malware samples because it can be invalidated by the recent tendency of required permissions. To be specific, recent malware samples tend to require unnecessary permissions to imitate benign apps.

In order to confirm the tendency, we investigated the number of permissions required by benign apps and malware samples. Fig. 1 shows distribution of the number of permissions required by benign apps and malware samples. In Fig. 1(a), 4,000 old malware samples are randomly selected from Drebin [16] (2010–2012). Meanwhile, in Fig. 1(b) and Fig. 1(c), 4,000 recent apps are randomly selected from Androzoo [17] (2017–2020) and VirusShare [18] (2018–2020), respectively. Furthermore, 4,000 benign apps are also randomly selected from Androzoo (2017–2020) for every figure in Fig. 1(a), Fig. 1(b), and Fig. 1(c). As shown in Fig. 1(a), the distribution of malware is relatively similar to that of benign apps. In particular, most apps require less than 30 permissions. On the other hand, as shown in Fig. 1(b) and Fig. 1(c), the distribution of malware is widely spread compared with Fig. 1(a), which is old dataset. In particular, many malware samples tend to require more than 30 permissions. These investigation results demonstrate that recent malware samples tend to require unnecessary permissions compared with old ones. In addition, we also investigated top ten permission pairs in our datasets. After that, we discovered that top ten permission pairs in benign apps also tend to be required by recent malware because of these tendencies, which causes degradation in detection performance of PermPair. Table 2 shows top ten permission pairs and their frequencies in Drebin, Androzoo, VirusShare, and the benign dataset. Furthermore, Table 3 shows the notation used for top permission pairs investigation. As shown in Table 2, out of ten permission pairs, there exist three common pairs, namely ANS:INT, INT:WES, and ANS:WES between the Drebin (2010–2012) and Benign apps. In other words, the permission pairs required by old malware in Drebin tend to be different from the ones of benign apps. On the other hand, six permission pairs are common among benign apps, Androzoo (2017–2020), and VirusShare (2018–2020). Moreover, their frequencies in Androzoo and VirusShare are higher than the ones in benign apps. For example, ANS:INT is the most

TABLE 2. Top ten permission pairs and their frequencies in Drebin, Androzoo, VirusShare, and the benign dataset. The pairs in gray cells are common ones.

Drebin (2010–2012)		Androzoo (2017–2020)		VirusShare (2018–2020)		Benign apps	
Permission pairs	Frequency	Permission pairs	Frequency	Permission pairs	Frequency	Permission pairs	Frequency
INT:RPS	0.908	**ANS:INT**	0.957	**ANS:INT**	0.981	**ANS:INT**	0.952
ANS:INT	0.697	INT:RPS	0.850	**INT:WES**	0.969	**INT:WES**	0.668
INT:WES	0.681	ANS:RPS	0.830	**ANS:WES**	0.961	**ANS:WES**	0.644
ANS:RPS	0.670	**INT:WES**	0.828	**AWS:INT**	0.941	INT:WL	0.530
RPS:WES	0.666	**ANS:WES**	0.810	INT:RPS	0.941	ANS:WL	0.521
ANS:WES	0.525	RPS:WES	0.774	**ANS:AWS**	0.940	**AWS:INT**	0.495
INT:SSMS	0.510	**AWS:INT**	0.666	ANS:RPS	0.936	**ANS:AWS**	0.493
INT:RBC	0.502	**ANS:AWS**	0.663	RPS:WES	0.935	WL:WES	0.421
RPS:RBC	0.491	AWS:RPS	0.632	**AWS:WES**	0.934	**AWS:WES**	0.403
RPS:SSMS	0.479	**AWS:WES**	0.623	AWS:RPS	0.917	INT:VR	0.371

TABLE 3. The notation used for top permission pairs analysis.

Permission	Notation
ACCESS_NETWORK_STATE	ANS
ACCESS_WIFI_STATE	AWS
INTERNET	INT
READ_PHONE_STATE	RPS
RECEIVE_BOOT_COMPLETED	RBC
SEND_SMS	SSMS
VIBRATE	VR
WAKE_LOCK	WL
WRITE_EXTERNAL_STORAGE	WES

frequently required pairs in benign apps, and its frequency is 0.952. Meanwhile, the frequencies are 0.957 and 0.981 in Androzoo and VirusShare, respectively. Similarly, as for five other pairs, their frequencies in Androzoo and VirusShare are also high compared with benign apps. In other words, it is natural that recent malware samples have permission pairs frequently required by benign apps, which means that more pairs appear in both benign apps and malware. As a result, apps cannot be correctly classified even through the frequencies are used. This is why we argue that detection schemes should not utilize the frequencies of permission pairs anymore. In particular, we discovered that PermPair [14], which is the latest scheme is considerably subject to this tendencies. PermPair regards permission pairs with high frequencies as important ones. Frequency based databases in PermPair can be contaminated when recent malware samples are utilized to create them. As a result, in most pairs, since the frequencies in malware tend to be higher than those in benign apps, the frequency based scores get ineffective. Although PermPair can meet "Efficiency" and "Intelligibility", PermPair cannot deal with recent malware.

Therefore, to adopt permission pairs based detection for practical use, we must devise a novel scheme that satisfies the three above requirements simultaneously.

V. PROPOSED SCHEME

In order to satisfy the above requirements, in this paper, we propose Android malware detection based on the CR of permission pairs The CR is defined as a ratio of a permission pair to all pairs in an app. We focus on the fact that the CR tends to be small in malware compared with benign apps. This is because malware samples tend to require unnecessary permissions to imitate benign apps. Our scheme calculates CRs of permission pairs from four aspects, namely Android Permission (AP), Dangerous Permission (DP), Custom Permission (CP), and all permissions. Although more than 300 permissions are predefined in Android specifications [19], our scheme regards the predefined permissions other than DPs as APs. DPs consist of 24 permissions that enable accessing sensitive data such as personal information in users' devices and are also specified in the specifications. CPs are permissions defined by developers while they are not in the specifications. All permissions are made up of all the above permissions, namely APs, DPs, and CPs. In our scheme, except for pairs generated from all permissions, pairs are composed only of the same kind of permissions. For example, if an app requires two APs (AP1 and AP2), three DPs (DP1, DP2, and DP3), and no CP, an AP based pair (AP1:AP2) and three DP based ones (DP1:DP2, DP2:DP3, and DP3:DP1) are created. In this case, the CRs based on APs and DPs are $1.0 = \frac{1}{1}$ and $0.33 = \frac{1}{3}$, respectively. As for the CR based on CP in this app, 0.0 is specially assigned to it because there is no CP based pair. Meanwhile, the CR based on all permission is $0.1 = \frac{1}{10}$ because there exist 10 pairs in total. Thus, the CRs based on APs, DPs, CPs, and all permissions are 1.0, 0.33, 0.0, and 0.1, respectively. The CRs are common among the same type of pairs. For instance, 0.33 is assigned to every DP based pair.

To leverage the tendencies of the CR for detection, four CRs based databases are constructed for each label of training databases (benign apps and malware). In other words, eight databases are prepared in total. For each app, we calculate eight similarity scores on the basis of corresponding databases. By doing this, even if malware requires unnecessary permissions, useful features of them can be obtained from a new aspect except for the frequency. Thus, our scheme can satisfy "Stability". Finally, as features in ML, our scores are fed into classifiers such as Random Forest (RF) [20], Support Vector Machine (SVM) [21], and Adaboost [22] for detection. Since our scheme utilizes just eight-dimensional features, it can realize reducing computational cost of training and compatibility with other features, which results in satisfaction of "Efficiency". Besides, our scores can quantita-

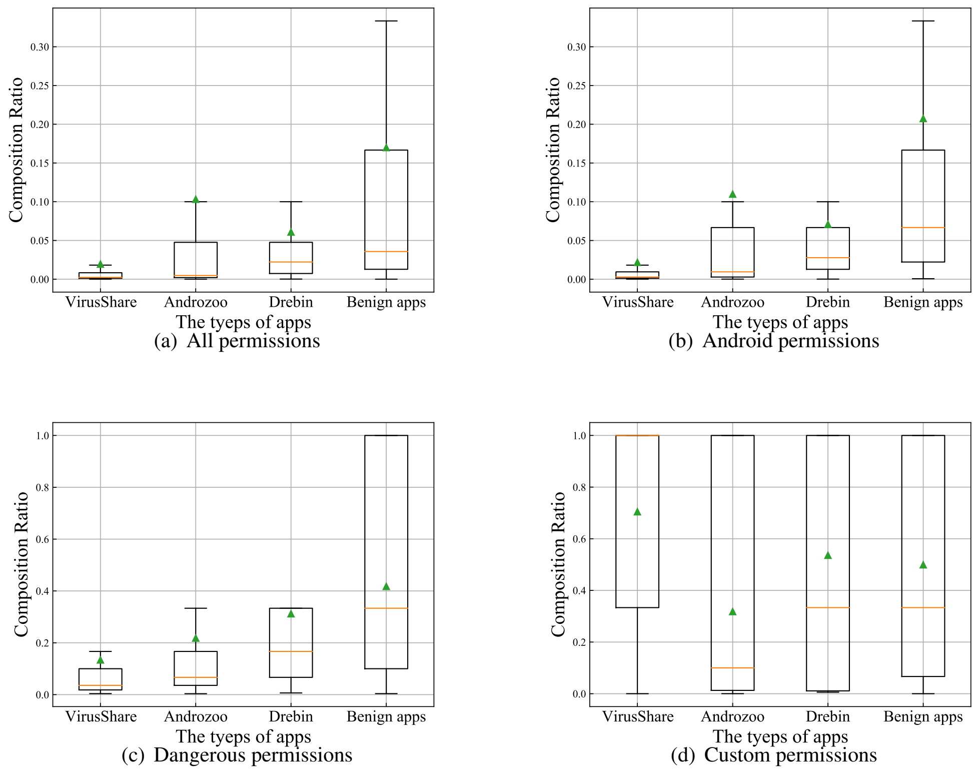


FIGURE 2. Box plots of the CR of permission pairs required by benign apps and malware samples. The orange lines and green triangles mean the median and the mean, respectively.

tively offer clear information that what types of permission pairs dictate detection results. Thus, since understandable information are provided for human, the proposed scheme also meets "Intelligibility". Since all the requirements can be met, our scheme is more suitable for practical use.

In what follows, we first validate usefulness of the CR. After that, the overview of the proposed scheme and its algorithms are explained in detail.

A. VALIDATION OF USEFULNESS OF COMPOSITION RATIO OF PAIRS

To validate that the CR of permission pairs are useful in malware detection, we investigated the distribution of the CR of permission pairs in apps by using box plot. Fig. 2 shows the box plots of the CR of permission pairs required by benign apps and malware samples in three types of dataset. In this investigation, 4,000 malware samples were randomly selected from each of Drebin [16], Androzoo [17], and VirusShare [18] datasets. Similarly, 4,000 benign apps were also randomly selected from benign dataset provided by Androzoo. In Fig. 2, there exist four box plots in terms of four types of permissions. As shown in Fig. 2(a), Fig. 2(b), and Fig. 2(c), in terms of all permissions, AP, and DP, there exist clear differences of the distribution between benign apps and malware samples. The CRs of malware samples from VirusShare, Androzoo, and Drebin are distributed within a range of 0.0 to 0.4. In particular, the variance of the CRs of malware samples from VirusShare is very small, which means that they tend to require more permissions. Moreover, it can be seen that the CRs in Drebin are also distributed in a narrow range, which means the CR is effective in detecting old malware. Meanwhile, the CRs of benign apps are more widely distributed than those of malware samples in the three types of permissions. Average value in benign apps are also higher than the ones in malware sources. With regard to CP, as shown in Fig. 2(d), the CRs are widely distributed in all types of apps. Therefore, obvious difference is not observed compared with three other types of permissions. However, since there exist differences of the median and the mean between two types of apps to some extent, we concluded that the CRs of CP based pairs can also be effective in detection. These results demonstrate that the CRs of permission pairs are useful to distinguish benign apps from malware samples. From these results, we concluded that using the CRs of four types of permission pairs are competent to correctly detect malware samples without relying on their frequencies.

B. ALGORITHM

In this section, we describe the algorithm of the proposed scheme. Fig. 3 shows the overview of the proposed scheme. The proposed scheme has two main phases, namely, 1) CR based database construction, 2) the proposed score calcula-

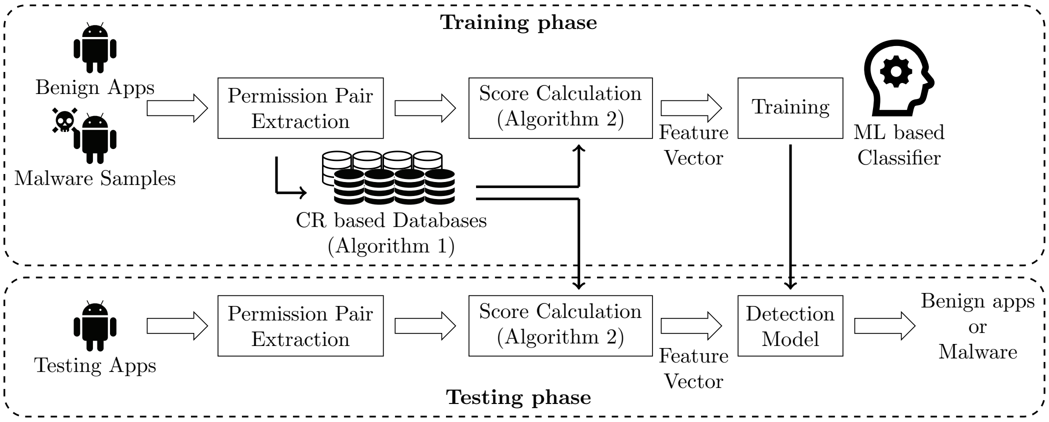


FIGURE 3. The overview of the proposed scheme.

tion. In the first phase, for each label, CR based databases are constructed depending on permission types. To capture more detailed features of apps, the proposed scheme constructs the four databases from the perspective of APs, DPs, CPs, and all permissions for each label. In the second phase, for each app, eight scores are calculated on the basis of corresponding databases constructed in the first phase. Finally, the eight scores are used as features for ML. The detailed algorithm of each operation is presented in the next section.

1) Composition Ratio Based Database Construction

Algorithm 1 shows CR based database construction. As an input, Algorithm 1 needs a set $TP_{pt,l} = \{P^{pt,l}_{ta_1}, ..., P^{pt,l}_{ta_i}, ..., P^{pt,l}_{ta_N}\}$, which consists of $P^{pt,l}_{ta_i}$ which is a set of permissions required by ta_i. Note that ta_i means an app in the training dataset, and the total number of ta_i is N. Furthermore, pt and l denote a type out of four permission types and a label of a training dataset, respectively. Algorithm 1 outputs a CR based database depending on pt and l. Let $D^{pt,l}$ denote an output database. If an input is a set of APs required by benign apps, $D^{\text{AP},\text{ben}}$ which means a database of CRs based on AP pairs for benign apps is constructed. For each ta_i, $P^{\text{pair}}_{ta_i}$ which means a set of permission pairs in ta_i is created on the basis of $P^{pt,l}_{ta_i}$. After that, each pair $(p_j, p_k) \in P^{\text{pair}}_{ta_i}$ and $cr_{(p_j,p_k)}$ which is the CR of pairs are registered with $D^{pt,l}$ as a key and a value, respectively. If (p_j, p_k) has already existed in $D^{pt,l}$, $cr_{(p_j,p_k)}$ are added to existing values that corresponds to (p_j, p_k). Finally, every value in $D^{pt,l}$ are divided by the total number of the training dataset, $|TP_{pt,l}|$, which means calculating the average CR in the training dataset. Similarly, Algorithm 1 is repeatedly conducted so as to create databases that corresponds to other permission types. Finally, four types of databases are constructed for each label, which means that eight databases are obtained in total.

2) Proposed Score Calculation

The proposed scores are calculated for every app in both a training dataset and testing one so as to create feature

Algorithm 1 CR Based Database Construction

```
Input: $TP_{pt,l} = \{P^{pt,l}_{ta_1}, ..., P^{pt,l}_{ta_i}, ..., P^{pt,l}_{ta_N}\}$
Output: A database $D^{pt,l}$ of $pt$ and $l$
 1: for each $P^{pt,l}_{ta_i} \in TP_{pt,l}$ do
 2:     Create $P^{\text{pair}}_{ta_i} = \{(p_1,p_2), ..., (p_j,p_k)\}$ from $P^{pt,l}_{ta_i}$
 3:     for each $(p_j,p_k) \in P^{\text{pair}}_{ta_i}$ do
 4:         if $|P^{\text{pair}}_{ta_i}| \neq 0$ then
 5:             $cr_{(p_j,p_k)} = \frac{1}{|P^{\text{pair}}_{ta_i}|}$
 6:         else
 7:             $cr_{(p_j,p_k)} = 0$
 8:         end if
 9:         if $(p_j,p_k)$ exists in $D^{pt,l}$ then
10:             $D^{pt,l}_{(p_j,p_k)} += cr_{(p_j,p_k)}$
11:         else
12:             $D^{pt,l}_{(p_j,p_k)} = cr_{(p_j,p_k)}$
13:         end if
14:     end for
15: end for
16: Divide every value in $D^{pt,l}$ by $|TP_{pt,l}|$
17: return $D^{pt,l}$
```

vectors. Algorithm 2 shows the proposed score calculation. In Algorithm 2, two scores, namely Malicious Score (MS) $MS^{pt}_{a_i}$ and Benign Score (BS) $BS^{pt}_{a_i}$ are calculated for an app a_i depending on pt. Note that $MS^{pt}_{a_i}$ and $BS^{pt}_{a_i}$ indicate a similarity score with malware and benign apps, respectively. The larger each score is, the higher the degree of similarity is. For each app a_i, eight scores are calculated in total. As an input, Algorithm 2 requires $P^{pt}_{a_i}$ that denotes a set of permissions in a_i. First of all, a set of permission pairs, $P^{\text{pair}}_{a_i} = \{(p_1,p_2), ..., (p_j,p_k), ..., (p_{N'-1}, p_{N'})\}$ is created from $P^{pt}_{a_i}$. Note that N' means the number of permissions in a_i. After that, $cr_{(p_j,p_k)}$ is compared with a value in $D^{pt,\text{mal}}$ and $D^{pt,\text{ben}}$. Here, $D^{pt,\text{mal}}$ represents a CR based database of malware in terms of pt. Similarly, $D^{pt,\text{ben}}$ means the database of benign apps. If (p_j,p_k) exists only in $D^{pt,\text{mal}}$, a similarity value for malware, $s_{\text{mal}} = 1$. Meanwhile, if (p_j,p_k) exists only in $D^{pt,\text{ben}}$, a benign similarity value,

Algorithm 2 The Proposed Score Calculation

```
Input: $P_{a_i}^{pt} = \{p_1, ..., p_j, ..., p_{N'}\}$ of an app $a_i$
Output: $MS_{a_i}^{pt}$ and $BS_{a_i}^{pt}$
1:  Create $P_{a_i}^{\mathrm{pair}} = \{(p_1, p_2), ..., (p_j, p_k), ..., (p_{N'-1}, p_{N'})\}$
2:  for each $(p_j, p_k) \in P_{a_i}^{\mathrm{pair}}$ do
3:      Initializing similarity values $s_{\mathrm{mal}} = 0$, $s_{\mathrm{ben}} = 0$
4:      if $|P_{ta_i}^{\mathrm{pair}}| \neq 0$ then
5:          $cr_{(p_j,p_k)} = \frac{1}{|P_{a_i}^{\mathrm{pair}}|}$
6:      else
7:          $cr_{(p_j,p_k)} = 0$
8:      end if
9:      if $(p_j, p_k)$ exists only in $D^{pt,\mathrm{mal}}$ then
10:         $s_{\mathrm{mal}} = 1$
11:     else if $(p_j, p_k)$ exists only in $D^{pt,\mathrm{ben}}$ then
12:         $s_{\mathrm{ben}} = 1$
13:     else if $(p_j, p_k)$ is both in $D^{pt,\mathrm{ben}}$ and $D^{pt,\mathrm{mal}}$ then
14:         $s_{\mathrm{mal}} = 1 - |cr_{(p_j,p_k)} - D_{(p_j,p_k)}^{pt,\mathrm{mal}}|$
15:         $s_{\mathrm{ben}} = 1 - |cr_{(p_j,p_k)} - D_{(p_j,p_k)}^{pt,\mathrm{ben}}|$
16:     end if
17:     if $s_{\mathrm{mal}} > s_{\mathrm{ben}}$ then
18:         $MS_{a_i}^{pt} += s_{\mathrm{mal}}$
19:     else if $s_{\mathrm{mal}} < s_{\mathrm{ben}}$ then
20:         $BS_{a_i}^{pt} += s_{\mathrm{ben}}$
21:     end if
22: end for
23: return $MS_{a_i}^{pt}$ and $BS_{a_i}^{pt}$
```

$s_{\mathrm{ben}} = 1$. In the case where (p_j, p_k) exists both in $D^{pt,\mathrm{mal}}$ and $D^{pt,\mathrm{ben}}$, s_{mal} and s_{ben} are calculated on the basis of absolute values of differences between $cr_{(p_j,p_k)}$ and statistical values in the databases, namely $|cr_{(p_j,p_k)} - D_{(p_j,p_k)}^{pt,\mathrm{mal}}|$ and $|cr_{(p_j,p_k)} - D_{(p_j,p_k)}^{pt,\mathrm{ben}}|$, respectively. The smaller absolute values are, the higher s_{mal} and s_{ben} are, which means high similarities. Finally, $MS_{a_i}^{pt}$ and $BS_{a_i}^{pt}$ are calculated depending on magnitude relationship of s_{mal} and s_{ben}. If s_{mal} is higher than s_{ben}, s_{mal} is added to $MS_{a_i}^{pt}$; otherwise s_{ben} is added to $BS_{a_i}^{pt}$. Similarly, $MS_{a_i}^{pt}$ and $BS_{a_i}^{pt}$ are calculated on the basis of other types of permission pairs. Thus, the eight scores are obtained for a_i. Finally, they are fed into a ML based classifier as feature vectors for training and testing.

VI. EVALUATION RESULTS AND DISCUSSION

A. DATASET

Table 4 shows datasets of APK files used in our simulation. Benign apps were obtained from Androzoo [17] distributed between 2017 and 2020 (2017–2020). Malware samples were collected from Drebin (2010–2012) [16], Androzoo (2017–2020), and VirusShare (2018–2020) [18]. In our evaluation, we used only the apps that require more than one permissions including all permissions because our work focus on appraising permission pairs based schemes. With regard to Androzoo and VirusShare, as a testing dataset, we randomly selected 1,500 benign apps and 1,500 malware samples from all of the benign dataset and all of the malware dataset, respectively. Besides, we constructed the training dataset by randomly selecting 6,000 benign apps and 6,000 malware samples from the rest of them. On the other hand, in terms of malware samples in the Drebin dataset, 1,000 apps and 3,000 apps are randomly selected as testing malware samples and training ones, respectively. This is because the total number of Drebin malware samples in our dataset is 4364. As for benign apps, 1,000 testing dataset and 3,000 training one are also randomly selected to balance the number of datasets.

TABLE 4. Datasets of APK files used in our evaluation.

	Source	Period	Testing	Training
Malware	Drebin [16]	2010–2012	1,000	3,000
	Androzoo [17]	2017–2020	1,500	6,000
	VirusShare [18]	2018–2020	1,500	6,000
Benign apps	Androzoo [17]	2017–2020	1,500	6,000
Benign apps for Drebin	Androzoo [17]	2017–2020	1,000	3,000

B. COMPARISON WITH CONVENTIONAL SCHEMES

1) Detection Performance

To evaluate the detection performance against old malware samples and recent ones, we evaluate ACCuracy (ACC), True Positive Rate (TPR), and False Positive Rate (FPR) with real dataset. ACC, TPR, and FPR are calculated as

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}, \tag{1}$$

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{2}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}, \tag{3}$$

where TP, TN, FP, and FN denote the number of True Positive (malware samples are regarded as malware samples), True Negative (benign apps are regarded as benign ones), False Positive (benign apps are regarded as malware samples), and False Negative (malware samples are regarded as benign apps), respectively.

We compare the proposed scheme with two conventional permission pairs based schemes, namely Boolean Vector Based Scheme (BVBS) [13] and PermPair [14]. Note that in [13], although both a single permission and permission pairs are used for detection, we only evaluate results by using permission pairs. This is because our focus is how to devise more practical detection with permission pairs. In our evaluation, BVBS utilizes top 40 permissions which frequently appear in benign apps and malware samples in the training datasets in keeping with the number of permissions in [13]. In other words, BVBS utilizes 780 permission pairs as features for ML. Furthermore, since it was reported that the RSBS [12] results in 55% FP with permission pairs, that scheme is not adopted for comparison in our evaluation.

In terms of the proposed schemes, we evaluate two types of schemes. The first one is a scheme with Random Forests

TABLE 5. Detection performance of the conventional schemes and the proposed schemes.

Scheme	Drebin (2010–2012)			Androzoo (2017–2020)			VirusShare (2018–2020)			Mixed dataset		
	ACC (%)	TPR (%)	FPR (%)	ACC (%)	TPR (%)	FPR (%)	ACC (%)	TPR (%)	FPR (%)	ACC (%)	TPR (%)	FPR (%)
Prop. (RF)	**97.3**	97.7	3.1	**80.4**	80.0	19.2	91.2	90.3	7.8	84.0	82.4	14.3
Prop. (SEL)	97.2	97.5	3.1	80.2	79.8	19.3	**91.3**	90.3	7.7	**84.2**	81.4	12.9
PermPair [14]	89.1	97.8	19.6	50.1	99.9	99.6	50.0	99.9	99.8	50.1	99.9	99.6
BVBS [13]	96.8	96.9	3.3	79.7	78.9	19.4	91.2	90.5	8.0	83.5	81.9	14.9

(RF) [20] (hereinafter, this scheme is called "Prop. (RF)"). The reason why we adopt RF is that RF achieved fast training and more accurate detection performance compared with other classifiers in our preliminary experiments. To fairly evaluate performance, BVBS also utilizes the RF classifier in our evaluation. The other is a scheme that uses a technique called Stacking Ensemble Learning (SEL) for training of ML (This is called "Prop. (SEL)"). The SEL can improve predictive performance by combining outputs from multiple classification models. In other malware detection methods [23], [24], the SEL is used to improve performance.

Table 5 shows detection performance of the conventional schemes and the proposed schemes. In Table 5, ACC, TPR, and FPR are appraised on every source. Furthermore, we evaluate detection performance of the four above schemes on a mixed dataset, which is composed of 6,000 benign apps and 6,000 malware samples from all sources. Note that the benign apps and malware samples are randomly selected as with other datasets. As shown in Table 5, the proposed schemes achieves the highest ACC compared with PermPair and BVBS. Moreover, the ACC of Prop. (SEL) is slightly higher than that of Prop. (RF), which means that SEL is useful for improving detection performance. In terms of PermPair, although the ACC on Drebin is 89.1%, the ACC values are 50.1% and 50.0% on Androzoo and VirusShare, respectively. In particular, FPRs are 99.6% and 99.8% in Androzoo and VirusShare, respectively, which means that almost all the benign apps are misjudged as malware. This is because PermPair is vulnerable to injecting unnecessary permissions in recent malware samples. Recent malware samples tend to require permission pairs that are frequently required by benign apps, as shown in Section IV. As a result, the frequencies of most pairs in malware tend to be higher than those in benign apps. These results show that frequency based databases get ineffective. Therefore, PermPair cannot correctly identify benign apps.

With regard to BVBS, it can achieve high detection performance as with Prop. (RF) and Prop. (SEL) whereas ACC is slightly low on all the four types of datasets. Since boolean vectors express information about the presence of pairs by using boolean values (0 or 1), each pair is equally treated regardless of its frequency. Hence, BVBS is insusceptible to the tendencies of recent malware, and its detection performance is not degraded. This evaluation demonstrates that BVBS and the proposed scheme are capable to precisely classify apps into benign apps and recent malware, which means that they satisfy "Stability". Although BVBS can fulfill "Stability", it

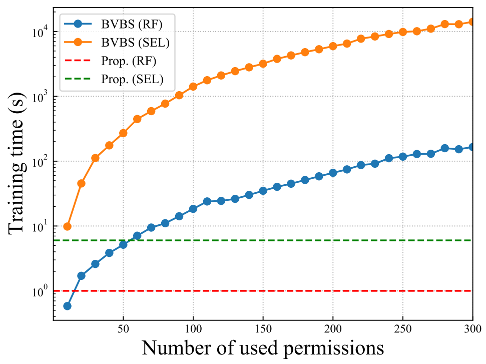


FIGURE 4. The training time in the proposed methods and BVBS when the number of used permissions are changed.

cannot satisfy the other requirements. In what follows, we compare the proposed scheme with BVBS in terms of aspects of "Efficiency" and "Intelligibility".

2) Computational Cost and Compatibility

As explained in Section VI-B 1), BVBS is tolerant to the tendency of recent malware samples. However, BVBS requires a high-dimensional feature vector to detect malware. In this subsection, we first compare the proposed scheme with BVBS in terms of training time. We ran our experiments on a machine with 2GHz Intel Core i5 and 16GB of RAM. After evaluating training time, we discuss compatibility with other features in ML based schemes. Through the comparison and discussion, we show that only the proposed scheme satisfies "Efficiency".

We evaluate relationship between the number of used permissions and training time. Fig. 4 shows the training time in the proposed methods and BVBS when the number of used permissions are changed. In this evaluation, 6,000 benign apps and 6,000 malware samples are used in BVBS (RF), BVBS (SEL), Prop. (RF), and Prop. (SEL). As shown in Fig. 4, training time of BVBS (RF) and BVBS (SEL) exponentially increase as the number of permissions increases. In particular, it takes up to 14,101s (= 3h 55m) for training of 12,000 apps when 300 permissions are used for detection in BVBS (SEL). In other words, BVBS (SEL) takes about 2,350 times longer than Prop. (SEL) in which training time is 6s.

Meanwhile, Fig. 5 shows the ACC of the proposed methods and BVBS when the number of used permissions are

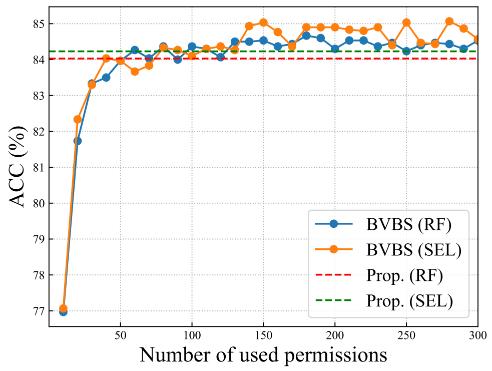


FIGURE 5. The ACC of the proposed methods and BVBS when the number of used permissions are changed.

changed. In Fig. 5, dotted lines of Prop. (RF) and Prop. (SEL) are just displayed as baselines. As shown in Fig. 5, as the number of permissions increases, ACC values of BVBS and BVBS (SEL) are slightly increased whereas the improvement is trivial. When 80 permissions are used in BVBS (3160 pairs are used), the ACC of BVBS and that of BVBS (SEL) are 84.36% and 84.33%, respectively, and these values are a little bit higher than that of Prop. (SEL), which is 84.2%. However, in this case, BVBS must utilizes 3160-dimensional vectors. On the other hand, since our features are eight-dimensional, the proposed scheme reduces the feature dimensions by about 99% with maintaining comparable detection accuracy. In the case where 280 permissions are used, whereas ACC of BVBS (SEL) is the best, already mentioned above, its computational cost of training gets huge. In this case, the number of permission pairs is 39,060, and training time is 13,007s (= 3h 36m). This result means that it is difficult to apply the SEL to BVBS in the practical situation although the SEL can ameliorate detection performance.

On the other hand, the training time of Prop. (RF) and that of Prop. (SEL) are 1s and 6s, respectively, in our evaluation. Training time of the proposed scheme is independent of the number of used permission pairs because our feature vector is always eight-dimensional one regardless of the number of pairs. In our environment, the average elapsed time for calculating eight scores per an app is about 9.3×10^{-4}s. In other words, our scheme can efficiently convert information about permission pairs into low-dimensional features.

Furthermore, because of the low-dimensional features, the proposed scheme is compatible with other ML based schemes. In other words, our scheme is promising for hybrid detection with feature fusion. The feature fusion means combining various features obtained from different aspects such as network traffic, API calls, ICC, and permissions so as to comprehensively detect various malware. The feature fusion is frequently adopted in Android malware detection [2], [9], [25], [26]. In [2], [26], permissions based features are combined with other features, and it is reported that the feature fusion is effective. When the feature fusion is carried out, the dimensions of feature vectors inevitably get large. For example, in [8], since more than 5,000-dimensional features can be used, it is desirable that additional features are low-dimensional ones. If the dimensions get too large like BVBS, detection performance may not be versatile because of curse of dimensionality. In particular, in the practical use, detecting malware comprehensively on the basis of various aspects is an ideal situation. Thus, in light of such situations, our eight scores are effective and desirable features, which means that the proposed scheme is suitable for a practical detection system with the feature fusion unlike BVBS. From these results, we concluded that the proposed scheme can satisfy not only “Stability” but also “Efficiency”.

C. ANALYSIS OF DETECTION RESULTS

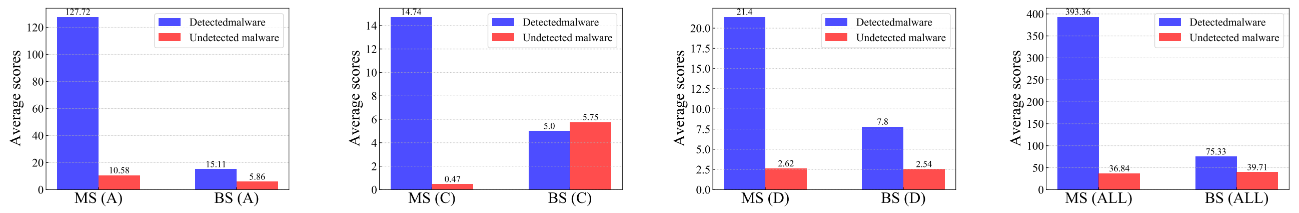
In this section, we demonstrate that our scores can offer useful information to understand detection results from a new aspect, namely the CR. In other words, we show that the proposed scheme can satisfy “Intelligibility”. Our scheme can provide understandable information about detection results because permission pairs based information is converted into eight simple scores on the basis of the CR based databases. The proposed scores and the CR based databases can help human to understand detection results as with PermPair. On the other hand, it is difficult for BVBS to provide understandable information about detection results to human by using its feature vector. In the following subsections, we conduct detailed analysis and interpretation for tested apps that our scheme judged.

1) Analysis for Detected Malware

In this subsection, we conduct analysis of malware samples that are judged by the proposed scheme. Fig. 6 shows average proposed scores of detected malware samples and undetected ones. Note that these average scores are calculated on the basis of 3,000 testing data, namely 1,500 benign apps and 1,500 malware samples from the mixed dataset. As shown in Fig. 6, there exist clear differences of our scores between detected malware samples and undetected ones. In terms of detected malware samples (blue bars), the differences between MS and BS tend to be large. In particular, with regard to scores of malware in Fig. 6(d), average MS (ALL) and BS(ALL) are 393.36, 75.33, respectively. The difference between them is 318.03. These results show that the proposed scheme can calculate useful scores for identifying malware samples.

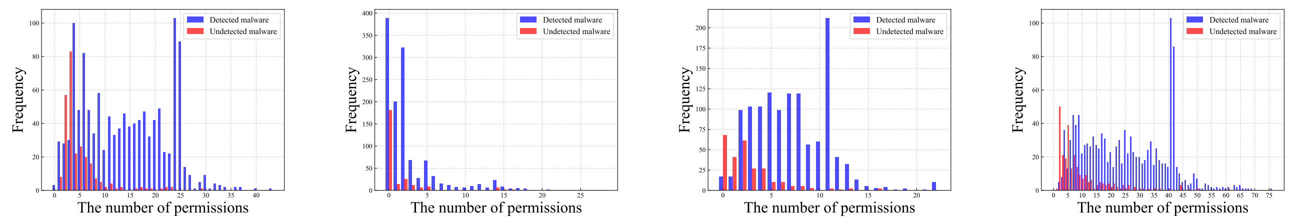
2) Analysis for Undetected Malware

On the other hand, as for undetected malware samples (red bars), it turns out that the differences between MS and BS are small compared with detected malware. These small differences cause the proposed scheme to misjudge them as benign apps. Through further analysis, we found the reason why these differences are small. That is, misjudged malware samples tend to require fewer permissions compared with malware samples that are correctly judged. As a result, since



(a) Android permissions (b) Custom permissions (c) Dangerous permissions (d) All permissions

FIGURE 6. Average proposed scores of detected malware samples and undetected ones. MS and BS mean Malicious Score and Benign Score, respectively. In terms of an alphabet and a word in parentheses after MS and BS, A, C, D, and ALL represent Android, Custom, Dangerous, and All permissions, respectively.



(a) Android permissions (b) Custom permissions (c) Dangerous permissions (d) All permissions

FIGURE 7. Distribution of the number of permissions required by detected malware samples and undetected ones.

CRs of permission pairs get large, they affect our scores. Fig. 7 shows distribution of the number of permissions required by detected malware samples and undetected ones. The used data are the same as the data in Fig. 6. As shown in Fig. 7, it can be seen that misjudged malware samples require fewer permissions in every type of permission. In terms of CPs, as shown in Fig. 7(b), even detected malware samples do not frequently require many CPs. As for APs and DPs, malware samples tend to require many of them, as shown in Fig. 7(a) and Fig. 7(c). Meanwhile, misjudged malware samples require fewer permissions. Since benign apps tend to require only essential permissions, CRs of them tend to be large. Thus, the CRs of misjudged malware samples are similar to those of benign apps, which results in incorrect scores of them.

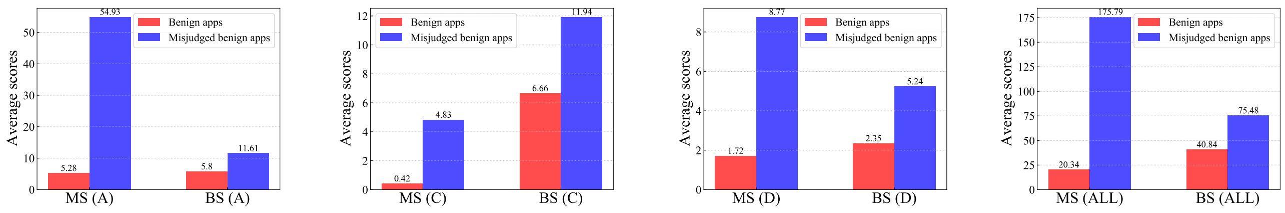
We found a misjudged malware sample that has five permissions in total. The malware has one CP and one DP, which means that permission pairs cannot be generated in the CP and the DP. Since the proposed scheme is based on permission pairs, it cannot calculate MS (C), BS(C), MS (D), and BS(D) at all. This malware requires INTERNET (INT) and WRITE_EXTERNAL_STORAGE (WES). This pair is dangerous, because it enables the malware to install another malware in a device. However, as for the pair INT:WES, s_{mal} and s_{ben} are 0.847 and 0.872, respectively, which means that benign features are more strong. Besides, although the CR based scores can be calculated in terms of APs and all permissions, both MS (A) and MS (ALL) were 0 whereas BS (A) and BS (ALL) were 2.42 and 5.29. As a result, since the two BSs are higher than the two MSs, the malware was misjudged as benign apps. Although this type of malware may dynamically require additional permissions, our scheme cannot deal with that. The above analysis reveals that our scheme has a limitation. We elaborate on limitations in Section VII.

3) Analysis for Benign Apps

In this subsection, we conduct analysis of benign apps that are judged by the proposed scheme. Fig. 8 shows average proposed scores of benign apps and misjudged ones. As shown in Fig. 8, in terms of benign apps (red bars), BSs are higher than MSs. In particular, the difference between MS (C) and BS (C) is the largest, which is 6.24, as shown in Fig. 8(b). The reason why scores of benign apps are relatively small is that benign apps tend to require fewer permissions. Fig. 8 shows distribution of the number of permissions required by benign apps and misjudged ones. As shown in Fig. 8(d), even when counting all permissions, the distribution tends to concentrate in fewer number. Thus, since the number of created pairs is also small, MSs and BSs tend to be small, which causes their differences to be small. However, it is can be seen that our scores are properly calculated because average BSs are larger than average MSs in benign app, which results in high detection performance.

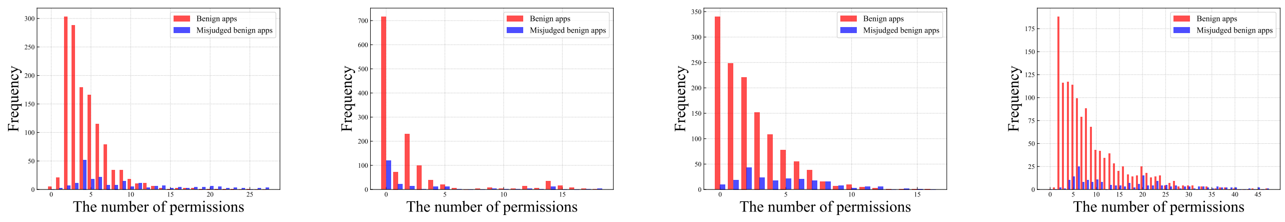
4) Analysis for Misjudged Benign Apps

On the other hand, as shown in Fig. 8(a), Fig. 8(c), and Fig. 8(d), magnitude correlation between MSs and BSs in misjudged benign apps (blue bars) are reversed compared with those in benign apps (red bars) except for MS (C) and BS (C) in Fig. 8(b). From this result, it turns out that CPs are less important than other permissions. Furthermore, the differences between MSs and BSs in misjudged benign apps are large compared with those in benign apps. Thus,



(a) Android permissions (b) Custom permissions (c) Dangerous permissions (d) All permissions

FIGURE 8. Average proposed scores of benign apps and misjudged ones.



(a) Android permissions (b) Custom permissions (c) Dangerous permissions (d) All permissions

FIGURE 9. Distribution of the number of permissions required by benign apps and misjudged ones.

these large differences causes misjudgement in the proposed scheme. Fig. 9 shows distribution of the number of permissions required by benign apps and misjudged ones. As shown in Fig. 9(a), Fig. 9(c) and Fig. 9(d), misjudged benign apps is liable to require more permissions. In this case, CRs of permission pairs in such benign apps are similar to the ones in malware because CRs get small. Hence, the proposed scheme regards the benign apps as malware by mistake when many permissions are required by benign apps. In order to prevent this types of misjudgement, it is important that developers of benign apps should avoid requiring unnecessary permissions. On the other hand, attackers have no choice but to inject unnecessary permission so as to conceal malicious evidence in permissions. Considering this fact, misjudgement by our scheme can be prevented by obeying the right manner of development in terms of permissions.

Meanwhile, the proposed scheme also failed in judging benign apps that require fewer permissions whereas the tendency of the CR of permission pairs is relatively similar to the benign one. To reveal the reason, we investigated such benign apps. After our investigation, we discovered that a benign app requires six permissions including three DPs, namely SEND_SMS (SSMS), READ_SMS (RSMS), and RECEIVE_SMS (RECSMS). MS (D) and BS (D) are 2.03 and 0.0, respectively. In addition to these DPs, this benign app also has INTERNET. By combining this permission and the above DPs, this benign app is potentially capable to conduct malicious operations. In fact, in terms of the pair INT:SSMS, similarities, s_{mal} and s_{ben} are 0.9349 and 0.9336, respectively, which means that this benign app is more similar to malware. As for other pairs such as INT:RSMS and INT:RECSMS, we found that s_{mal} is larger than s_{ben}. As a result, MS (ALL) and BS (ALL) are 11.2 and 2.83, respectively. To cope with this misjudgement, using other features such as API calls and ICC is needed.

VII. LIMITATION

Our scheme can satisfy requirements for practical use. However, there exist some limitations even in the proposed scheme. The first one is that the proposed scheme cannot deal with malware samples that statically require fewer permissions, as explained in Section VI-C 2). This limitation is common to all the permission based schemes that utilize static analysis [11]–[14]. In order to address this limitation, extracting dynamic permissions [27] is promising. We plan to devise a new scheme that calculates the CR from pairs of permissions including dynamic ones. However, in order to obtain dynamic permissions, it is necessary to run each app in a device or an emulator, which causes runtime overhead. Since dynamic analysis based schemes can also capture more useful features such as network traffic and API calls, the feature fusion may be more useful. In light of such facts and the compatibility of our low-dimensional features, combining our features and other features can be promising for detecting the above malware.

Furthermore, as mentioned in Section VI-C 4), our scheme cannot correctly judge benign apps that require fewer permissions including DPs. Such benign apps might actually need to use DPs for benign purposes. It is difficult to justify the truth on the basis of information about permissions. Thus, we concluded that such benign apps should be treated by using other feature based schemes.

At this stage, our scheme utilizes all permission pairs required by apps. As a result, detection performance of the proposed scheme is slightly inferior to BVBS with pairs of more than 80 permissions. In order to ameliorate the performance, we might have to select useful pairs from all pairs. It is possible that using only pairs whose variance of the CR is small is helpful in improving detection performance. We will work on this task as future work.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we have proposed Android malware detection scheme using the CR of permission pairs. We focus on the fact that the CR tends to be small in malware because of unnecessary permissions. To leverage the CR for malware detection, we constructed databases regarding the CR. For each app, we calculated eight similarity scores on the basis of the prepared databases. Finally, our scores were fed into machine learning (ML) based classifiers for detection. By using real datasets, our evaluation results show that the proposed scheme can detect malware with up to 97.3% accuracy. Compared with BVBS, our scheme can reduce the feature dimensions by about 99% with maintaining comparable detection accuracy on recent datasets. Because of this, the proposed scheme is compatible with other features in ML based schemes. Furthermore, our features can quantitatively offer clear information that what types of permission pairs dictate detection results. Thus, our scheme is more suitable for practical use. However, our scheme has the limitations. To address them, our future work will focus on devising a new scheme that calculates the CR from pairs of permissions including dynamic ones. Besides, in order to improve detection performance, we plan to select useful pairs by using some techniques such as clustering CRs with their variance in future.

REFERENCES

[1] "Smartphone market share, 2020." Accessed: May 2021. [Online]. Available: https://www.idc.com/promo/smartphone-market-share/os.

[2] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, "Madam: Effective and efficient behavior-based android malware detection and prevention," IEEE Transactions on Dependable and Secure Computing, vol. 15, no. 1, pp. 83–97, 2016.

[3] Z. Wang, Q. Liu, and Y. Chi, "Review of android malware detection based on deep learning," IEEE Access, vol. 8, pp. 181 102–181 126, 2020.

[4] S. Wang, Q. Yan, Z. Chen, B. Yang, C. Zhao, and M. Conti, "Detecting android malware leveraging text semantics of network flows," IEEE Transactions on Information Forensics and Security, vol. 13, no. 5, pp. 1096–1109, 2018.

[5] S. Garg, S. K. Peddoju, and A. K. Sarje, "Network-based detection of android malicious apps," International Journal of Information Security, vol. 16, no. 4, pp. 385–400, 2017.

[6] Y. Aafer, W. Du, and H. Yin, "Droidapiminer: Mining api-level features for robust malware detection in android," in International conference on security and privacy in communication systems. Springer, 2013, pp. 86–103.

[7] L. Deshotels, V. Notani, and A. Lakhotia, "Droidlegacy: Automated familial classification of android malware," in Proceedings of ACM SIGPLAN on program protection and reverse engineering workshop 2014, 2014, pp. 1–12.

[8] K. Xu, Y. Li, and R. H. Deng, "Iccdetector: Icc-based malware detection on android," IEEE Transactions on Information Forensics and Security, vol. 11, no. 6, pp. 1252–1264, 2016.

[9] H. Cai, N. Meng, B. Ryder, and D. Yao, "Droidcat: Effective android malware detection and categorization via app-level profiling," IEEE Transactions on Information Forensics and Security, vol. 14, no. 6, pp. 1455–1470, 2018.

[10] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, P. G. Bringas, and G. Álvarez, "Puma: Permission usage to detect malware in android," in International Joint Conference CISIS'12-ICEUTE 12-SOCO 12 Special Sessions. Springer, 2013, pp. 289–298.

[11] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an, and H. Ye, "Significant permission identification for machine-learning-based android malware detection," IEEE Transactions on Industrial Informatics, vol. 14, no. 7, pp. 3216–3225, 2018.

[12] S. Liang and X. Du, "Permission-combination-based scheme for android mobile malware detection," in 2014 IEEE international conference on communications (ICC). IEEE, 2014, pp. 2301–2306.

[13] X. Liu and J. Liu, "A two-layered permission-based android malware detection scheme," in 2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering. IEEE, 2014, pp. 142–148.

[14] A. Arora, S. K. Peddoju, and M. Conti, "Permpair: Android malware detection using permission pairs," IEEE Transactions on Information Forensics and Security, vol. 15, pp. 1968–1982, 2019.

[15] W. Wang, X. Wang, D. Feng, J. Liu, Z. Han, and X. Zhang, "Exploring permission-induced risk in android applications for malicious application detection," IEEE Transactions on Information Forensics and Security, vol. 9, no. 11, pp. 1869–1882, 2014.

[16] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket." in Ndss, vol. 14, 2014, pp. 23–26.

[17] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "Androzoo: Collecting millions of android apps for the research community," in Mining Software Repositories (MSR), 2016 IEEE/ACM 13th Working Conference on. IEEE, 2016, pp. 468–471.

[18] "virusshare.com," Accessed: Apr. 2020. [Online]. Available: https://virusshare.com.

[19] "Manifest.permission," Accessed: July. 2021. [Online]. Available: https://developer.android.com/reference/android/Manifest.permission?hl=ja.

[20] L. Breiman, "Random forests," Machine learning, vol. 45, no. 1, pp. 5–32, 2001.

[21] C. Cortes and V. Vapnik, "Support-vector networks," Machine learning, vol. 20, no. 3, pp. 273–297, 1995.

[22] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," Journal of computer and system sciences, vol. 55, no. 1, pp. 119–139, 1997.

[23] D. Vasan, M. Alazab, S. Venkatraman, J. Akram, and Z. Qin, "Mthael: Cross-architecture iot malware detection based on neural network advanced ensemble learning," IEEE Transactions on Computers, vol. 69, no. 11, pp. 1654–1667, 2020.

[24] H. Zhu, Y. Li, R. Li, J. Li, Z.-H. You, and H. Song, "Sedmdroid: An enhanced stacking ensemble of deep learning framework for android malware detection," IEEE Transactions on Network Science and Engineering, 2020.

[25] G. Tao, Z. Zheng, Z. Guo, and M. R. Lyu, "Malpat: Mining patterns of malicious and benign android apps via permission-related apis," IEEE Transactions on Reliability, vol. 67, no. 1, pp. 355–369, 2017.

[26] A. Arora and S. K. Peddoju, "Ntpdroid: a hybrid android malware detector using network traffic and system permissions," in 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). IEEE, 2018, pp. 808–813.

[27] A. Mahindru and P. Singh, "Dynamic permissions based android malware detection using machine learning techniques," in Proceedings of the 10th innovations in software engineering conference, 2017, pp. 202–210.

HIROYA KATO was born in Gunma, Japan in 1994. He received his B.E. and M.E. degrees from Keio University, in 2017 and 2019, respectively, where he is currently pursuing the Ph.D. degree. His research interest is security & privacy for IoT. He is a member of IEICE.

TAKAHIRO SASAKI was born in Saitama, Japan in 1995. He received his B.E. degrees from Keio University in 2021. His research interest is security & privacy for IoT.

IWAO SASASE was born in Osaka, Japan in 1956. He received the B.E., M.E., and D.Eng. degrees in Electrical Engineering from Keio University, Yokohama, Japan, in 1979, 1981 and 1984, respectively. From 1984 to 1986, he was a Post Doctoral Fellow and Lecturer of Electrical Engineering at the University of Ottawa,ON, Canada. He is currently a Professor of Information and Computer Science at Keio University, Yokohama, Japan. His research interests include modulation and coding, broadband mobile and wireless communications, optical communications, communication networks and information theory. He has authored more than 301 journal papers and 446 international conference papers. He granted 48 Ph.D. degrees to his students in the above field. Dr. Sasase received the 1984 IEEE Communications Society (ComSoc) Student Paper Award (Region 10), 1986 Inoue Memorial Young Engineer Award, 1988 Hiroshi Ando Memorial Young EngineerAward, 1988 Shinohara MemorialYoung EngineerAward, 1996 Institute of Electronics, Information, and Communication Engineers (IEICE) of Japan Switching System Technical Group Best Paper Award, and WPMC2008 Best Paper Award. He is now serving as a Vice-President of IEICE. He served as President of the IEICE Communications Society (2012-2014). He was Board of Governors Member-at-Large (2010-2012), Japan Chapter Chair (2011-2012), Director of the Asia Pacific Region (2004-2005), Chair of the Satellite and Space Communications Technical Committee (2000-2002) of IEEE ComSoc., Vice President of the Communications Society (2004-2006), Chair of the Network System Technical Committee (2004-2006), Chair of the Communication System Technical Committee (2002-2004) of the IEICE Communications Society, Director of the Society of Information Theory and Its Applications in Japan (2001-2002). He is Fellow of IEICE, and Senior Member of IEEE, Member of the Information Processing Society of Japan.