Cross-Layer Profiling of Encrypted Network Data for Anomaly Detection

Fares Meghdouri
TU Wien
Vienna, Austria
fares.meghdouri@tuwien.ac.at

Félix Iglesias Vázquez
TU Wien
Vienna, Austria
felix.iglesias@tuwien.ac.at

Tanja Zseby
TU Wien
Vienna, Austria
tanja.zseby@tuwien.ac.at

***Abstract*—In January 2017 encrypted Internet traffic surpassed non-encrypted traffic. Although encryption increases security, it also masks intrusions and attacks by blocking the access to packet contents and traffic features, therefore making data analysis unfeasible. In spite of the strong effect of encryption, its impact has been scarcely investigated in the field. In this paper we study how encryption affects flow feature spaces and machine learning-based attack detection. We propose a new cross-layer feature vector that simultaneously represents traffic at three different levels: *application*, *conversation*, and *endpoint behavior*. We analyze its behavior under TLS and IPSec encryption and evaluate the efficacy with recent network traffic datasets and by using Random Forests classifiers. The cross-layer multi-key approach shows excellent attack detection in spite of TLS encryption. When IPsec is applied, the reduced variant obtains satisfactory detection for botnets, yet considerable performance drops for other types of attacks. The high complexity of network traffic is unfeasible for monolithic data analysis solutions, therefore requiring cross-layer analysis for which the multi-key vector becomes a powerful profiling core.**

***Index Terms*—Network Data Analysis, Encrypted Communications, Anomaly Detection, Machine Learning.**

I. Introduction

Network communications are progressively moving toward encryption. According to a study by Cisco Systems [1], encrypted Internet traffic surpassed non-encrypted traffic in January 2017. In particular, in 2019, web traffic reached encryption rates above 80% [1]. When focusing on network attacks, security experts foresee that more than 70% of malware will use some kind of encryption over 2020, whereas about 60% of worldwide organizations will fail to decrypt HTTPS efficiently, becoming vulnerable to encrypted threats [1].

In addition to other issues related to computational requirements and data privacy, encryption forces network security defenses to detect attacks based on contextual flow data [2], i.e., indirectly capturing malicious traffic by finding behavioral patterns in protocol headers and aggregated flow information. This approach has become the modern paradigm for attack detection in network communications [3]. However, it is important to remark that the selection of features for attack detection and network traffic classification is still an open question in the scientific community. This problem is exacerbated by the fact that the implications of encryption in feature selection processes or in the presentation of novel detection vectors are commonly forgotten or obviated.

In this work we propose a novel flow-feature vector that combines different traffic representations that have shown satisfactory performance results in previous works. Our proposal is named *multi-key* as it condenses three alternative ways of modeling network traffic: *application*, *conversation*, and *endpoint* profiles. We additionally study the compatibility with the two most utilized protocols for encrypting modern Internet traffic at network and transport layers: TLS and IPsec. We leave apart QUIC since it is still in a development phase, specifications change continuously, and the technology is not consolidated yet. For the evaluation of our approach, we run experiments over three recent IDS (Intrusion Detection System) datasets and use detection algorithms based on Random Forests (RF). Decision Tree-based learners are the supervised classifiers that tend to show the most accurate, robust, and explainable performances in the field, e.g., [4], [5], [6]. In addition, we use RF since they provide a better and simpler explanation of predictions compared to other supervised learning techniques based on Deep Learning (DL) and to compare performance with other studies.

Experiments show that when the *multi-key* vector is applied, attack detection is outstanding regardless of TLS encryption. The higher coverage of IPsec causes performance drops for any detector, but the *multi-key*-based approach still reaches satisfactory results when identifying botnets hidden under IPsec tunnels. Results support the triple perspective of the *multi-key* vector as a step forward in the design of new data representations with higher discriminant power, and therefore an excellent baseline for future IDS.

Summarizing, the main contributions of this paper are:

- Introducing the *multi-key* vector, a novel cross-layer representation of network traffic data that profiles traffic from *application*, *conversation*, and *endpoint behavior* perspectives.
- Testing the *multi-key* vector for attack detection with three realistic IDS datasets published in 2014, 2015, and 2017, and compare its performance with previous works.
- Studying the impact of TLS and IPsec encryption in network attack detection when using the *multi-key* vector to represent traffic flows.

The remainder of the paper is organized as follows. In Section II, we revisit literature about flow-based attack detection. In Section III, main concepts related to the analysis and encryption in network traffic are introduced. In Section IV, we explain the proposed vector for representing network traffic and discuss how it deals with encryption. In Section V, we provide a step-by-step description of the experimental setup. In Section VI, results from experiments are shown and discussed. Conclusions are given in Section VII.

II. Related Work

Key features are essential requisites for solving any analytical problem. In other words, without proper features you cannot conveniently dissect input spaces in a way that the aimed classes are clearly isolated. In [7] Domingos expresses this idea with regard to machine learning. He states that "determining which features to use is the most important factor of a successful machine learning algorithm". As for the intersection between machine learning and network traffic analysis, Lim et al. [8] emphasize this point as the first of three main challenges, which are: (1) key feature selection, (2) finding the best analysis algorithms, and (3) obtaining representative datasets for training and testing algorithms. In the same work, Lim et al. insist on the relevance of investigating the discriminative power of features. However, the variety of features that can be used for flow analysis is immense. A quick look on the IPFIX flow information elements defined by IANA is overwhelming [9], and this only shows a fraction of the many possibilities. Also, there is no formal knowledge that specifically links known attacks with flow/contextual/behavioral features or profiles, but for well-known, obvious attack classes (e.g., scanning, DoS), and perhaps for the case of *packet length*, which has been recognized as a key feature for the discrimination of attacks in different works [10], [8], [11], [12].

Several studies focus on the impact of features for attack detection and traffic classification. For example, the 41 base features present in the most popular and frequently used IDS datasets for the last two decades (Darpa-KDD'98, Darpa-KDD'99, and NSL-KDD [13]) are studied in [14]. Their discriminant power and extraction costs are discussed, concluding with a reduced subset of 16 outstanding features. This set of features might be nowadays deemed as incidental due to the fact that network traffic technologies, protocols, uses, and threats have highly evolved and changed since these datasets were collected and generated. In this regard, in 2005, Moore et al. provide a complete list of 248 features to be used for traffic classification and attack detection [15]. Such list collects all the features that have been traditionally used in the mentioned datasets and in the most cited research in the field. As mentioned before, the availability of such features in encrypted environments is not discussed in the cited report due to the relatively recent adoption of encryption in network communications.

Recently, some works have delved into the implications of encryption from analytical perspectives. In [5] Velan et al. provide a survey of methods for encrypted traffic classification and analysis, but mostly from an application layer level. An overview of deep learning for encrypted traffic analysis is presented in [16], covering works published in the last three years. Reviewed works successfully apply Generative Adversarial Networks (GAN), Autoencoders (AE), Convolutional Networks (CNN) and Long-Short Term Memories (LSTM) for protocol identification and traffic classification, but only one is reported for intrusion detection: [17]. However, the cited work evaluates its method with the old Darpa'98 and ISCX2012 datasets, which are not encrypted (or, if there is any encryption, this is within the application layer). In general, last years research on deep learning and advanced machine learning for encrypted traffic consider encryption at application levels and mostly for identifying protocols and applications (e.g., [18]), also obviating the discussion about features or assuming the capability of neural networks to extract features. In this regard, it is worth emphasizing authors' reasoning in [5] when they state: "we also discovered that the use of information from the unencrypted parts of encrypted connections for a network anomaly detection is only briefly investigated by researchers", indirectly pointing to the study of available features (statistics, headers).

In line with this observation, the concern about features for attack detection has been recently tackled in [11], also including discussions about the implications of encryption. In this work, five representations are studied and compared using the UNSW-NB15 dataset [6]. Authors conclude that host-based profiling [19] has the best performance and a summary vector obtained from meta-analysis [20] gives the best complexity-accuracy tradeoff.

The *multi-key* vector is built by performing a novel optimal combination of the following feature vectors despite their distinct nature:

- The vector proposed by Williams et al. in 2006 [21], referred to as CAIA, which has been frequently used (with minimal variations) in traffic classification and attack detection with machine learning.
- The TA vector [22], used to identify main patterns in backbone Internet traffic according to unidirectional, straightforward time series footprints.
- The AGM vector [19], designed for discovering profiles in the Internet Background Radiation and consisting on observing network devices rather than flows.
- The Consensus vector [20], that uses a meta-analytical approach and establishes the recommended vector after weighting the features used in the most prominent research conducted for the last 15 years.

Noteworthy is the work by Anderson and McGrew in [2]. Authors proposed a fine grained representation with very low error rates for detecting malwares in TLS-encrypted traffic. The vector is highly customized with around 650 features extracted with the Cisco/Joy tool.[1] Unfortunately, many of the used features are not available in most IDS datasets, fact that either complicates its use in comparative studies or significantly downgrades its performance [23].

[1]github.com/cisco/joy

III. Network Traffic Encryption

In this section we introduce the studied encryption protocols for network and transport layers.

A. *TLS (Transport Layer Security)*

TLS [24] is a cryptographic protocol that provides confidentiality and data integrity on top of the transport layer. It is a commonly used protocol; for instance, with HTTPS to offer secure web browsing. During the TLS handshake a secure connection is established after which encrypted data is transmitted.

Since TLS is running on top of the transport layer, all fields of the IP and TCP headers remain unencrypted. Furthermore, TLS encrypts all packets of a single TCP connection individually without tunneling and, hence, connections can be analyzed separately. In TLS 1.2, some parts of the initial handshake are transmitted unencrypted (e.g., signature exchange for mutual authentication), which allows further analysis during the connection establishment. However, in TLS 1.3, several steps of the initial handshake are already encrypted, which further reduces analysis possibilities. The Datagram Transport Layer Security (DTLS) protocol [25] is a variant of TLS that provides security features for datagram protocols (such as UDP). The encrypted packets look similar to the TLS packets, but DTLS still needs to deal with packet loss and reordering by adding sequence numbers.

B. *IPsec (Internet Protocol Security)*

IPsec [26] is an end-to-end security protocol at the network layer. Security Associations (SAs) are established using the Internet Key Exchange protocol (IKE). IPsec operates in different modes: transport mode and tunnel mode.

In transport mode the endpoints of the security association are hosts. IPsec adds an Encapsulating Security Payload (ESP) header before the transport header and everything after the ESP header is encrypted. This means that transport header fields cannot be used for traffic analysis. In IPv6 the ESP is implemented as an extension header, also encrypting all subsequent extension headers. In addition, IPsec usually aggregates all traffic from one endpoint into one SA. However, it is possible to select which packets are encrypted at the endpoint, but typically all traffic from the source of the SA to the destination of the SA is encrypted, therefore aggregating multiple application flows from a host or a router. This aggregation of several application flows into a larger `source-destination` flow further impairs analysis possibilities.

In tunnel mode, the IPsec SA is established between two routers, meaning that traffic from the hosts to the router remains unencrypted (e.g., because the local networks are trusted). The router later establishes a tunnel and encrypts all the traffic. Here, the original IP header (with the IP addresses of source and destination host) is also encrypted and a new IP header with the IP addresses of the router endpoints are added instead. In this case, the original host IP addresses cannot be recovered from the encrypted traffic. Since the tunnel can be used by multiple hosts, the level of aggregation is even higher than in transport mode. The tunnel mode also works with a mixed setup with a host and a router as endpoints. One specific example is the establishment of a VPN. In such a case the SA starts at a host (which, for instance, visits a foreign untrusted network) and ends at a gateway (which is the entry point to the trusted home network). In this case, the new header contains the original host IP address as source IP address and the entry router of the home network as destination IP address.

This work only covers IPsec encryption in transport mode.

IV. Multi-Key Feature Vector

Previous works suggest that the combination of different feature vectors might lead to enhanced attack detection performances [11]. We explore this idea by bringing together diverse flow-based representations in the literature. We obviate the ones that are obsolete, use obscure or unpublished extraction methods/models, use features that are too costly or non-clearly defined, demand deep packet inspection, or require information not available in the data due to network restrictions. Therefore, we reduce the scope and build a new mixed representation based on CAIA, TA, AGM and Consensus feature vectors (introduced in Section II, features shown in the Appendix, Table IV). Constructing the cross-layer vectors has implications that we discuss as follows.

A. *Combining traffic representations*

Merging different traffic vectors is not a straightforward task. The CAIA, TA, AGM, and Consensus formats use different flow-keys (see Table I), some being uni-directional and others bi-directional. To make such representations compatibles, we join information extracted from each variant by intersecting flow-keys in a rigorous way.

Key Feature	**CAIA & Cons.**	**TA**	**AGM$_s$**	**AGM$_d$**
sourceIPAddress	●	●	●	.
destinationIPAddress	●	●	.	●
sourceTransportPort	●	.	.	.
destinationTransportPort	●	.	.	.
protocolIdentifier	●	.	.	.

TABLE I: Flow keys of Consensus, CAIA, TA, AGM$_s$ (profiling sources), and AGM$_d$ (profiling destinations) vectors.

Note that, when flows are defined based on the traditional 5-tuple (“sourceIPAddress”, “destinationIPAddress”, “sourceTransportPort”, “destinationTransportPort”, “protocolIdentifier”), datasamples capture information about the *application*. Instead, if the 2-tuple (“sourceIPAddress”, “destinationIPAddress”) is used, datasamples profile *conversations* between two hosts. Finally, the 1-tuple (either “sourceIPAddress” or “destinationIPAddress”) implies datasamples collecting *endpoint behaviors*. In the *multi-key* feature vector, the 5-tuple is kept as a basis, and the information extracted using the 2-tuple and 1-tuple keys are subsequently added.

Table IV shows the features that form the *multi-key* vector. The AGM vector is used twice: taken “sourceIPAddress” as flow key (AGM$_s$) and taken “destinationIPAddress” as flow key (AGM$_d$). This allows profiling endpoints as information senders and receivers respectively. Profiling endpoints is very useful to identify some attack-related behavioral schemes; for instance, scanners or victims sending back-scatter (senders), or victims under Denial of Service (DoS) (receivers).

The example in Figure 1 can help understand how the *multi-key* vector is created. In the example, we can see a small network with three hosts —A, B, and C— that communicate with each other. After a predefined *observation time*, communications are captured in four different flows. Figure 1 focuses on flow number 4, which corresponds to the UDP *application* between A and B, started by A and with predefined port numbers. The traditional 5-tuple key is used as a "preliminary" key to extract CAIA and Consensus features. Note that the same vector additionally captures information related to the whole *conversation* between A and B (2-tuple, TA features) and therefore ignores the concept of port number. Finally, information about the *endpoint behaviors* of A as a sender and B as a receiver is also stored in the same flow vector which consists of the entire traffic going from or into A and B.

The power of the *multi-key* approach lies in collecting information related to the application, the conversation, and the behavior of endpoints at the same time and in the same sample; in other words, they store: single application statistics (CAIA and Consensus), conversation statistics (TA) and single host (attacker or victim) statistics (AGM). Detectors obtain a deeper insight of the analyzed traffic and can profile threats based on the nature of the specific application, the step-by-step strategy described in the conversation, the global behavior of an attacker or a victim in the network, or all these aspects at the same time. This allows to detect patterns that can not be otherwise detected by using each representation separately.

B. Reducing Redundancy

Since network data features are known to be highly correlated [14], merging different traffic representations is prone to involve a strong redundancy (multicollinearity or high feature correlation). Hence, after joining the different formats, feature selection is required to keep only relevant features. In this respect, RF have shown to be robust feature selection filters and able to evaluate feature importance during classification [27].

C. Dealing With Encryption

Encryption prevents extracting features from certain layers. For instance, with TLS, all features encapsulated above the transport layer are not accessible. Respectively, with IPsec in transport mode, features encapsulated above the network layer are not available either. This has some implications for the *multi-key* feature vector:

- No constraints for the TLS case, i.e., fully compatible.
- The 5-tuple flow-key cannot be built for the IPsec case. Additionally, all features extracted from the transport layer are unavailable. A reduced version of the *multi-key* vector only using the 2-tuple key is therefore used in this case.

The right column of Table IV (*Av*) indicates which features are available for the complete vector (not encrypted or TLS), and for the reduced variant (with IPsec).

V. Experiments

In this section, we describe the conducted experiments. We introduce the used datasets, the classification test-bed, and the evaluation metrics. Scripts are available for further exploration and replication in our repository[2].

A. Intrusion Detection Systems Datasets

- Proposed by Sharafaldin et al. [28], the CICIDS2017 dataset is one of the most recent datasets for IDS evaluation. In this dataset, normal traffic is generated with realistic user profiles, while up-to-date attacks are sequentially triggered in a supervised environment. Authors created the dataset following the eleven quality criteria discussed in [28]. Figure 2a shows the distribution of traffic instances, which includes seven attack families and a variety of benign traffic shapes.
- The UNSW-NB15 dataset by Mustafa et al. [29] contains a considerably larger variation of attack types that correspond to nine threat families. Both normal and malicious traffic are generated with the *IXIA PerfectStorm* platform.[3] Figure 2b shows the class distributions.
- The ISCX-Bot-2014 dataset by Beigi et al. [30] particularly focuses on botnets. It implements variations of 16 types of modern botnet techniques mixed with normal, harmless traffic instances. Figure 2c shows class distributions.

Figure 2 uses different colors —orange and blue— to represent the count of extracted flows when using each of the *multi-key* variations: complete (with TLS) and reduced (with IPsec). The IPsec variant contains fewer flows due to a more coarse-grained aggregation.

B. Classification Test-bed

In short, the experimental test-bed: (a) retrieves raw pcaps as inputs, (b) extracts flows in vectors according to pre-configured features, (c) joins multiple flow vectors to form the *multi-key* feature set, (d) labels flow-instances based on ground truth files (for evaluation purposes), (e) performs transformations, data splits, cross-validation, parameter tuning, feature selection, and data reduction, (f) analyzes preprocessed flows with supervised learners, and (g) evaluates classification results and provides performance results.

The experimental test-bed can be split in three main blocks:

1) Extraction.
   In this phase we used *go-flows*[4] to extract information from raw traffic captures (pcaps) into network flows.

[2]github.com/CN-TU/multi-key-vector-experiments
[3]ixiacom.com/products/perfectstorm
[4]github.com/CN-TU/go-flows

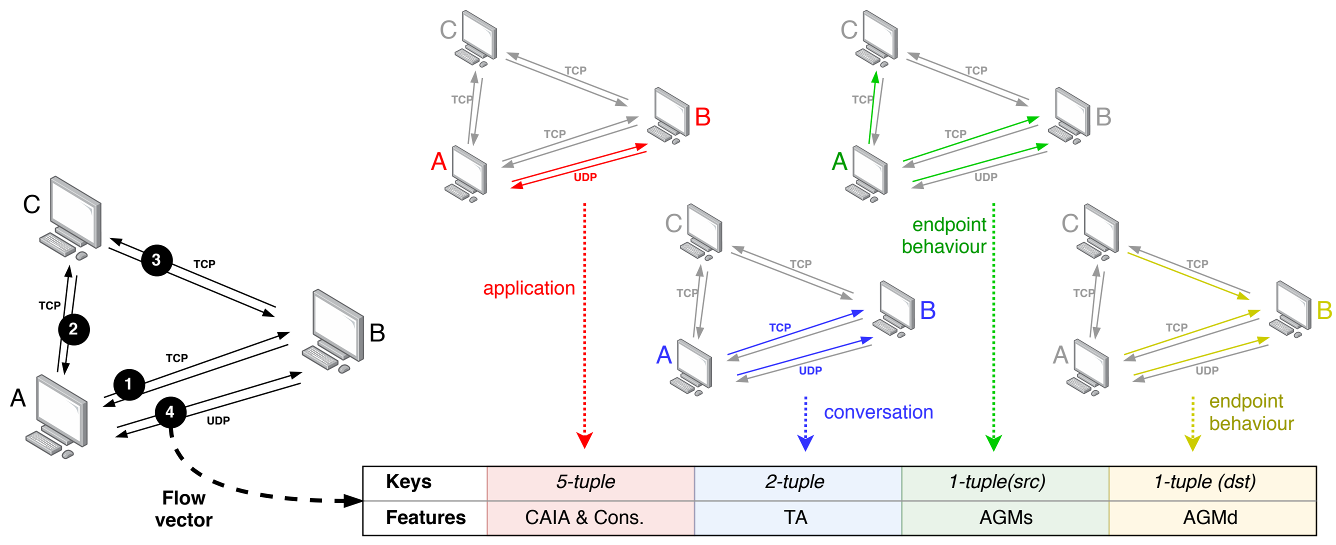
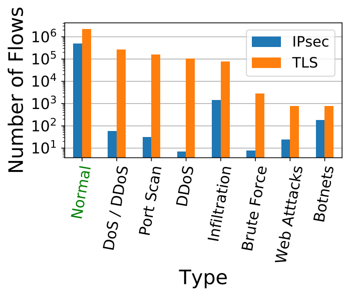
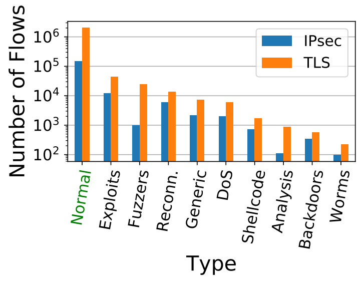


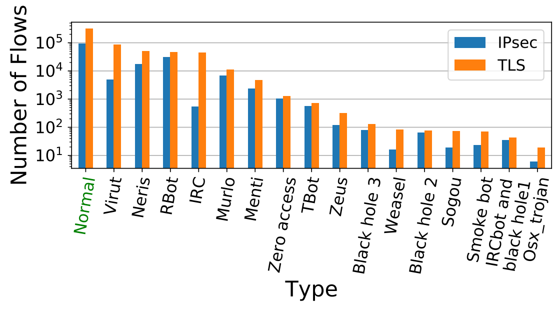
Fig. 1: Example of the *multi-key* construction. In the example network, hosts A, B, and C exchange packets through different applications. Given a flow vector (e.g., vector number 4), it contains information about the application (in red), the conversation (in blue), and the behaviors of the source and destination (in green and yellow respectively).



(a) CICIDS2017 dataset



(b) UNSW-NB15 dataset



(c) ISCX-Bot-2014 dataset

Fig. 2: Class distributions when extracting flow using both variants of the *multi-key* feature vector (TLS and IPsec). Class frequency is shown in a logarithmic scale.

Flows are built and joined together to form the *multi-key* vector introduced in Section IV. Finally, instances are labeled based on the ground truth files (provided by the data sources) to allow subsequent evaluation. To allow for further analysis, two kinds of labels are used: "attack name" (categorical) and "malicious-related/non-malicious-related" (binary).

2) Preprocessing.
   - *Train-test separation*. Data instances are split into a training set (70%) and a test set (30%). We applied stratified sampling to keep equivalent class proportions.
   - *Standardization*. Data is z-score normalized. This is required to optimize posterior Principal Component Analysis (PCA) transformations.
   - *Feature selection*. Irrelevant features (i.e., features not used during the classification) are removed with RF.
   - *Feature reduction*. We apply PCA to shrink the resulting space (whenever possible) with no information loss. This step makes analysis by RF faster and simpler [31].

3) Analysis.
   - *Parameter tuning* of RF with genetic algorithm search by maximizing the *F1* metric.
   - *Cross-validation*. Even in spite of the fact that RF naturally minimize overfitting, a 5-fold cross-validation step is performed to reinforce generalization.
   - *Evaluation*. Performance indices are obtained based on the metrics defined in Section V-C.

C. Evaluation metrics

We evaluate performances by using Roc-Auc and F1 scores for consistency with previous works. Some works have emphasized that the Prec-Rec (Precision-Recall) curve is better suited than the Roc curve when evaluating binary classification with unbalanced classes, e.g., [32], [33]. Therefore, we additionally provide the Average Precision (A-Precision), which can be computed from the Prec-Rec curve [34]. We refer readers to [34] and [35] for further information about the Prec-Rec curve and the Average Precision metric.

D. Explainability

In order to better understand the power of the *multi-key* feature as a profiling tool, we additionally analyzed features prior to space transformations by means of Shapley Additive Explanations (SHAP) [36], [37]. To do this, we used a tree explainer that computes *Shapley Values*, which represent the marginal contribution of each input feature to the final label. The tree explainer shows the average amount of change in the output when perturbing inputs. Such values allow understand decisions made by the classifier and enable detection of overfitting and data artifacts.

VI. Results and Discussion

In this section we discuss experimental results. We provide evaluations of the *multi-key* feature vector for TLS and IPsec encryption and compare them with previous works. All experiments are available for further tests and replicability in our website [2].

A. Detection Performances

Table II collects performance results when evaluating over the three IDS datasets under test with both encryption schemes TLS and IPsec. Some insights can be inferred:

- As a general rule, *the multi-key vector obtains high detection rates*. Moreover, the selection of RF, the use of cross-validation, and the fact that training and test results are consistent guarantee robustness and reduce possible overfitting. Cross-validation shows low variance, suggesting stability in the learning models regardless of the datasets.
- *IPsec encryption considerably affects attack detection in general, but not botnet detection in particular*. By comparing TLS and IPsec variants, performance drops occur for the IPsec case in the CICIDS2017 and ISCX-Bot-2014 datasets. This was expected because the IPsec encryption removes data that is potentially useful for the identification of attacks. The performance drop is more acute for the CICIDS2017 dataset, whereas it is minor for the ISCX-Bot-2014 case. This suggests that the detection of botnets is less affected by IPsec encryption than other types of attacks. A plausible explanation is that botnet flows are isolated in each IPsec tunnel and, therefore, the behavior captured by the *multi-key* vector is not disturbed.
- *The multi-key vector captured generation artifacts in the UNSW-NB15 dataset*. The IPsec case shows better performances than the TLS variant for the UNSW-NB15 dataset. After carefully checking this counter-intuitive result, we realized that this is caused by the design of the UNSW-NB15 dataset. The dataset seems to expect analysis from *application* perspectives (i.e., aggregations based on a 5-tuple key). Attacking addresses in the dataset perform many types of attacks, such behavior being very easy to spot when using the *endpoint* approach included in the *multi-key* representation (see Figure 4.d). This design aspect can be considered as a *defect* in the UNSW-NB15 IDS dataset, but also shows the benefits of using multi-perspective approaches for analyzing traffic.

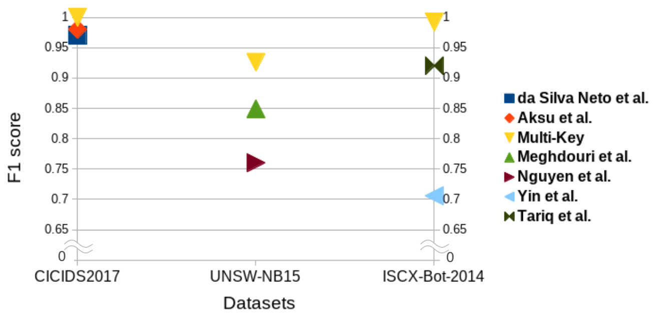


Fig. 3: Performance comparison based on the F1-score. da Silva Neto et al. [38], Aksu et al. [39], Meghdouri et al. [11], Nguyen et al. [40], Yin et al. [41], Tariq et al. [42].

Candidate	CICIDS2017	UNSW-NB15	ISCX-Bot-2014
Multi-key	1	0.926	0.992
da Silva Neto et al.	0.970	-	-
Aksu et al.	0.980	-	-
Meghdouri et al.	-	0.849	-
Nguyen et al.	-	0.760	-
Yin et al.	-	-	0.705
Tariq et al.	-	-	0.920

TABLE III: Performance comparison F1 scores.

In short, results show that the *multi-key* feature vector obtains satisfactory performances and outperforms previous research that uses the same datasets [28], [38], [11], [40], [41], [42] and research that uses the same feature vectors individually [11]. Figure 3 and Table III show a comparison among works published between 2017 and 2019.

B. Analysis of Features

Figure 4 shows the top 5-features used by the RF classifier to distinguish “attack-related” from “non-attack-related” flows. The figure shows six plots corresponding to the two studied encryptions (TLS and IPsec) and the three tested IDS datasets (ISCX-Bot-2014, UNSW-NB15, and CICIDS2017).

Results show that there is a higher coincidence in the discriminant features discovered for the ISCX-Bot-2014 and CICIDS2017, whereas the UNSW-NB15 shows a more different feature set. This is consistent with the generation artifacts observed in the UNSW-NB15 dataset (discussed in Section VI-A). Ideally, discovered feature sets should capture *necessary* patterns that are idiosyncratic of the used attack types, and not *contingent* mechanisms that belongs to the dataset generation setup. Therefore, coincidence in the top discriminant features through different IDS datasets is a good

TABLE II: Performance indices of conducted experiments.

CICIDS2017	MD	MSL	Accuracy	Precision	Recall	F1-score	A-Precision	Roc-Auc
$TLS - training$	30	1	1.000 ±0.000	1.000 ±0.000	0.999 ±0.000	0.999 ±0.000	1.000 ±0.000	1.000 ±0.000
$TLS - testing$			1.000 ±0.000	1.000 ±0.000	0.999 ±0.000	0.999 ±0.000	1.000 ±0.000	1.000 ±0.000
$IPsec - training$	30	1	0.999 ±0.000	0.997 ±0.008	0.792 ±0.074	0.882 ±0.048	0.927 ±0.013	0.984 ±0.005
$IPsec - testing$			0.999 ±0.000	0.992 ±0.021	0.758 ±0.046	0.859 ±0.037	0.882 ±0.034	0.963 ±0.017

UNSW-NB15	MD	MSL	Accuracy	Precision	Recall	F1-score	A-Precision	Roc-Auc
$TLS - training$	26	2	0.994 ±0.000	0.939 ±0.002	0.934 ±0.003	0.936 ±0.002	0.981 ±0.002	0.999 ±0.000
$TLS - testing$			0.994 ±0.000	0.929 ±0.007	0.931 ±0.009	0.930 ±0.003	0.973 ±0.002	0.999 ±0.000
$IPsec - training$	19	6	0.994 ±0.001	0.965 ±0.008	0.992 ±0.002	0.978 ±0.003	0.987 ±0.006	0.999 ±0.000
$IPsec - testing$			0.994 ±0.001	0.965 ±0.008	0.993 ±0.005	0.979 ±0.004	0.989 ±0.006	0.999 ±0.000

ISCX-Bot-2014	MD	MSL	Accuracy	Precision	Recall	F1-score	A-Precision	Roc-Auc
$TLS - training$	30	1	0.997 ±0.000	0.999 ±0.000	0.993 ±0.001	0.996 ±0.001	1.000 ±0.000	1.000 ±0.000
$TLS - testing$			0.994 ±0.001	0.998 ±0.000	0.989 ±0.001	0.993 ±0.001	1.000 ±0.000	1.000 ±0.000
$IPsec - training$	25	2	0.996 ±0.001	0.998 ±0.001	0.991 ±0.003	0.995 ±0.001	1.000 ±0.000	1.000 ±0.000
$IPsec - testing$			0.994 ±0.002	0.997 ±0.002	0.989 ±0.004	0.993 ±0.002	1.000 ±0.000	1.000 ±0.000

MD: `max-depth`, MSL: `min-samples-at-leaf`. The `number-of-estimators` is set to the half of number of features. Other parameters take default values according to the Scikit-learn implementation (`V0.20.2`).

sign, though differences are expected because attack collections in each dataset are different, harmless or non-attack-related traffic is also differently generated, and, as emphasized before, correlation in network traffic features is usually strong.

As for the specific features, we observe that revealing detection patterns mainly combine:

- The volume of data sent by the client (in the conversation: "_NTATData" and in the application "octetTotalCount"), both in absolute values and in time-rates.
- The value and number of different TTLs ("ipTTL"). TTL values are useful to identify operating systems.
- The number of distinct aimed destination IPs ("distinct (destinationIPAddress)"), which are key to identify scanners, machines performing active information gathering, or spreading malware.
- The most used TCP flag ("tcpFlags"), known to be essential components in specific attack schemes, e.g., TCP SYN Flood, Tsunami SYN Flood.
- The packet length ("ipTotalLength"), already considered as highly discriminant in previous works (Section II).
- The total number of packets ("packetTotalCount"). Highly correlated to the volume of sent data, it also characterizes the dynamics of the network traffic element regardless of such element being the application, the conversation, or the endpoint behaviour.
- The different used protocols ("protocolIdentifier"). Infiltration strategies commonly check out known vulnerabilities to enter the aimed machine, therefore involving various protocols. However, in the UNSW-NB15 case, it is likely disclosing a contingent pattern.

VII. Conclusion

In this work, we have introduced the *multi-key* feature vector, a new cross-layer approach to build network data representations for attack detection and traffic classification. The *multi-key* vector assembles state of the art feature vectors in a way that they are able to profile traffic at three different levels: application, conversation, and endpoint behavior. The *multi-key* vector draws more informative analysis spaces and leverages the fact that each type of network attack has a more suitable analysis perspective from which it can be better differentiated and detected.

We have tested our approach with three modern, last-generation IDS datasets (CICIDS2017, UNSW-NB15, and ISCX-Bot-2014), and studied the implications of IPsec and TLS encryption for the analysis. In general, results show enhanced detection when compared to previous works. The *multi-key* vector can maintain high detection performances in spite of TLS encryption, but it considerable losses power in traffic encrypted with IPsec for a variety of attacks. This does not include malicious botnets, which are satisfactorily identified also with IPsec encryption. We have studied the generated RF models using SHAP values, which show that the *multi-key* vector feature can adapt to different traffic classes, attack types, and encryption schemes.

Modern attack detection in network communications currently faces serious challenges related to the analysis of big streams of encrypted data. The selection of network traffic features is an open question that deserves high attention due to the need for secure communications irrespective of the volume, speed, and data privacy mechanisms. The *multi-key* feature vector is a complete, lightweight, and flexible form to represent network traffic communications that allows the accurate

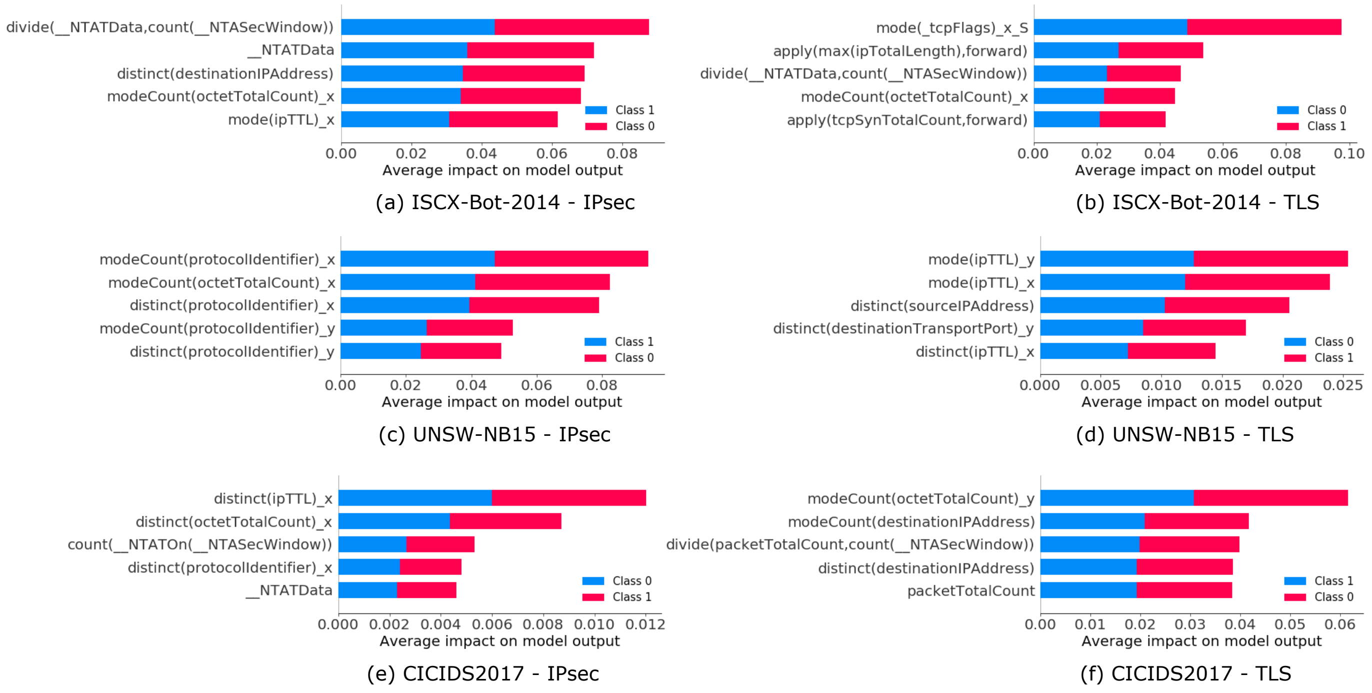


Fig. 4: The five most relevant features sorted by the average impact on the prediction based on SHAP values. We analyze the contributions with each variant/dataset separately to allow better understanding of the decision mechanism of the RF classifier.

profiling of traffic classes and attacks, therefore becoming the basis for effective machine-learning based network security and data analysis.

References

[1] Cisco and its affiliates, "White paper: Cisco encrypted traffic analytics," Cisco, Tech. Rep., 2019.

[2] B. Anderson and D. McGrew, "Identifying encrypted malware traffic with contextual flow data," in *Proceedings of the 2016 ACM workshop on artificial intelligence and security*. ACM, 2016.

[3] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An overview of IP flow-based intrusion detection," *IEEE Commun. Surveys Tutorials*, 2010.

[4] D. Qin, J. Yang, J. Wang, and B. Zhang, "Ip traffic classification based on machine learning," in *2011 IEEE 13th International Conference on Communication Technology*. IEEE, 2011.

[5] P. Velan, M. Čermák, P. Čeleda, and M. Drašar, "A survey of methods for encrypted traffic classification and analysis," *International Journal of Network Management*, 2015.

[6] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set," *Information Security Journal: A Global Perspective*, 2016.

[7] P. M. Domingos, "A few useful things to know about machine learning." *Commun. acm*, 2012.

[8] Y.-s. Lim, H.-c. Kim, J. Jeong, C.-k. Kim, T. T. Kwon, and Y. Choi, "Internet traffic classification demystified: on the sources of the discriminative power," in *Proceedings of the 6th International COnference*. ACM, 2010.

[9] B. Claise and B. Trammell, "Rfc 7012: Information model for ip flow information export (ipfix)," Internet Engineering Task Force (IETF), Tech. Rep., 2013.

[10] A. Este, F. Gringoli, and L. Salgarelli, "On the stability of the information carried by traffic flow features at the packet level," *SIGCOMM Comput. Commun. Rev.*, 2009.

[11] F. Meghdouri, T. Zseby, and F. Iglesias, "Analysis of lightweight feature vectors for attack detection in network traffic," *Applied Sciences*, 2018.

[12] L. Peng, H. Zhang, B. Yang, and Y. Chen, "Feature evaluation for early stage internet traffic identification," in *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 2014.

[13] Information and I. Computer Science, University of California. Kdd cup 1999 data. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[14] F. Iglesias and T. Zseby, "Analysis of network traffic features for anomaly detection," *Machine Learning*, 2015.

[15] A. Moore, D. Zuev, and M. Crogan, "Discriminators for use in flow-based classification," Queen Mary, University of London, Dept of Computer Science., Tech. Rep., 2005.

[16] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: An overview," *IEEE communications magazine*, 2019.

[17] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, pp. 1792–1806, 2018.

[18] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, 2017.

[19] F. Iglesias and T. Zseby, "Pattern discovery in internet background radiation," *IEEE Transactions on Big Data*, 2017.

[20] D. C. Ferreira, F. I. Vázquez, G. Vormayr, M. Bachl, and T. Zseby, "A meta-analysis approach for feature selection in network traffic research," in *Proceedings of the Reproducibility Workshop*. ACM, 2017.

[21] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification," *ACM SIGCOMM Computer Communication Review*, 2006.

[22] F. Iglesias and T. Zseby, "Time-activity footprints in ip traffic," *Computer Networks*, 2016.

[23] F. Iglesias, A. Hartl, T. Zseby, and A. Zimek, "Are network attacks outliers? a study of space representations and unsupervised algorithms," in *ECML-PKDD Workshops. Machine Learning for Cybersecurity (MLCS)*, 2019, publication pending.

[24] E. Rescorla, "The transport layer security (tls) protocol version 1.3," RFC, 2018.

[25] E. Rescorla and N. Modadugu, "Datagram transport layer security version 1.2," RFC, 2012.

[26] S. Frankel and S. Krishnan, "Ip security (ipsec) and internet key exchange (ike) document roadmap," RFC, 2011.

[27] Y. Saeys, T. Abeel, and Y. Van de Peer, "Robust feature selection using ensemble feature selection techniques," in *Machine Learning and Knowledge Discovery in Databases*, W. Daelemans, B. Goethals, and K. Morik, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 313–325.

[28] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." in *ICISSP*, 2018.

[29] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*. IEEE, 2015.

[30] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *2014 IEEE Conference on Communications and Network Security*. IEEE, 2014.

[31] J. Hu, J. Deng, and M. Sui, "A new approach for decision tree based on principal component analysis," in *2009 Int. Conf. on Computational Intelligence and Software Engineering*, 2009.

[32] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006.

[33] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets," *PloS one*, 2015.

[34] K. Boyd, K. H. Eng, and C. D. Page, "Area under the precision-recall curve: point estimates and confidence intervals," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2013.

[35] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, 2006.

[36] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017.

[37] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee, "From local explanations to global understanding with explainable ai for trees," *Nature Machine Intelligence*, 2020.

[38] M. G. da Silva Neto and D. G. Gomes, "Network intrusion detection systems design: A machine learning approach," in *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. SBC, 2019.

[39] D. Aksu, S. Üstebay, M. A. Aydin, and T. Atmaca, "Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature selection algorithm," in *International Symposium on Computer and Information Sciences*. Springer, 2018.

[40] K. K. Nguyen, D. T. Hoang, D. Niyato, P. Wang, D. Nguyen, and E. Dutkiewicz, "Cyberattack detection in mobile cloud computing: A deep learning approach," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2018.

[41] C. Yin, Y. Zhu, S. Liu, J. Fei, and H. Zhang, "An enhancing framework for botnet detection using generative adversarial networks," in *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*. IEEE, 2018.

[42] F. Tariq and S. Baig, "Machine learning based botnet detection in software defined networks," *International Journal of Security and Its Applications*, 2017.

APPENDIX

TABLE IV: Multi-key vector features. The Availability (Av) of each features is represented as follows: "T": feature available with TLS encryption, "I": feature available with IPsec encryption (transport mode), "*": feature removed before the analysis either because it shows a sparse distribution or it is irrelevant for the classification task. The feature naming is based on the IPFIX and IANA standards. A few features/functions are newly introduced: (1) "modeCount": it is the number of packets sent to the most recurring source or destination, (2) "__NTA{X}": features introduced in [22].

	Feature	Av
General features	flowStartMilliseconds	TI*
	flowDurationMilliseconds	TI
	sourceIPAddress	TI*
	destinationIPAddress	TI*
	sourceTransportPort	T*
	destinationTransportPort	T*
	protocolIdentifier	T
From CAIA & Consensus	apply(packetTotalCount,forward)	TI
	apply(octetTotalCount,forward)	TI
	apply(min(ipTotalLength),forward)	TI
	apply(max(ipTotalLength),forward)	TI
	apply(median(ipTotalLength),forward)	TI
	apply(mean(ipTotalLength),forward)	TI
	apply(mode(ipTotalLength),forward)	TI
	apply(stdev(ipTotalLength),forward)	TI
	apply(min(_interPacketTimeSeconds),forward)	TI
	apply(max(_interPacketTimeSeconds),forward)	TI
	apply(median(_interPacketTimeSeconds),forward)	TI
	apply(mean(_interPacketTimeSeconds),forward)	TI
	apply(variance(_interPacketTimeSeconds),forward)	TI
	apply(tcpSynTotalCount,forward)	T
	apply(tcpAckTotalCount,forward)	T
	apply(tcpFinTotalCount,forward)	T
	apply(_tcpCwrTotalCount,forward)	T
	apply(packetTotalCount,backward)	TI
	apply(octetTotalCount,backward)	TI
	apply(min(ipTotalLength),backward)	TI
	apply(max(ipTotalLength),backward)	TI
	apply(median(ipTotalLength),backward)	TI
	apply(mean(ipTotalLength),backward)	TI
	apply(mode(ipTotalLength),backward)	TI
	apply(stdev(ipTotalLength),backward)	TI
	apply(min(_interPacketTimeSeconds),backward)	TI
	apply(max(_interPacketTimeSeconds),backward)	TI
	apply(median(_interPacketTimeSeconds),backward)	TI
	apply(mean(_interPacketTimeSeconds),backward)	TI
	apply(variance(_interPacketTimeSeconds),backward)	TI

Continued on next column

Continued from previous column

	Feature	Av
	apply(tcpSynTotalCount,backward)	T
	apply(tcpAckTotalCount,backward)	T
	apply(tcpFinTotalCount,backward)	T
	apply(_tcpCwrTotalCount,backward)	T
From AGM(source/destination)	distinct(destinationIPAddress/sourceIPAddress)	TI
	mode(destinationIPAddress/sourceIPAddress)	TI*
	modeCount(destinationIPAddress/sourceIPAddress)	TI
	distinct(sourceTransportPort)	T
	mode(sourceTransportPort)	T*
	modeCount(sourceTransportPort)	T
	distinct(destinationTransportPort)	T
	mode(destinationTransportPort)	T*
	modeCount(destinationTransportPort)	T
	distinct(protocolIdentifier)	T
	mode(protocolIdentifier)	T
	modeCount(protocolIdentifier)	T
	distinct(ipTTL)	TI
	mode(ipTTL)	TI
	modeCount(ipTTL)	TI
	distinct(_tcpFlags)	T
	mode(_tcpFlags)	T
	modeCount(_tcpFlags)	T
	distinct(octetTotalCount)	TI
	mode(octetTotalCount)	TI
	modeCount(octetTotalCount)	TI
	packetTotalCount	TI
From TA	__NTAFlowID	T*
	__NTAProtocol	T
	__NTAPorts	T*
	__NTATData	TI
	packetTotalCount	TI
	count(__NTASecWindow)	TI
	divide(__NTATData,count(__NTASecWindow))	TI
	divide(packetTotalCount,count(__NTASecWindow))	TI
	max(__NTATOn(__NTASecWindow))	TI
	min(__NTATOn(__NTASecWindow))	TI
	max(__NTATOff(__NTASecWindow))	TI
	min(__NTATOff(__NTASecWindow))	TI
	count(__NTATOn(__NTASecWindow))	TI