Journal of
Sensor and Actuator Networks

MDPI

Article

An Approach for Stego-Insider Detection Based on a Hybrid NoSQL Database †

Igor Kotenko [1,*], Andrey Krasov [2], Igor Ushakov [2] and Konstantin Izrailov [1,2]

[1] St. Petersburg Federal Research Center of the Russian Academy of Sciences, 199178 St. Petersburg, Russia; konstantin.izrailov@mail.ru
[2] Department of Secured Communication Systems, The Bonch-Bruevich State University of Telecommunications, 199178 St. Petersburg, Russia; krasov@inbox.ru (A.K.); ushakovia@gmail.com (I.U.)
* Correspondence: ivkote@comsec.spb.ru

† This paper is an extended version of the conference paper: Igor Kotenko, Andrey Krasov, Igor Ushakov and Konstantin Izrailov. Detection of Stego-Insiders in Corporate Networks Based on a Hybrid NoSQL Database Model. The 4th International Conference on Future Networks and Distributed Systems (ICFNDS 2020). Saint-Petersburg, Russia, 26–27 November 2020.

Abstract: One of the reasons for the implementation of information security threats in organizations is the insider activity of its employees. There is a big challenge to detect stego-insiders-employees who create stego-channels to secretly receive malicious information and transfer confidential information across the organization's perimeter. Especially presently, with great popularity of wireless sensor networks (WSNs) and Internet of Things (IoT) devices, there is a big variety of information that could be gathered and processed by stego-insiders. Consequently, the problem arises of identifying such intruders and their transmission channels. The paper proposes an approach to solving this problem. The paper provides a review of the related works in terms of insider models and methods of their identification, including techniques for handling insider attacks in WSN, as well methods of embedding and detection of stego-embeddings. This allows singling out the basic features of stego-insiders, which could be determined by their behavior in the network. In the interests of storing these attributes of user behavior, as well as storing such attributes from large-scale WSN, a hybrid NoSQL database is created based on graph and document-oriented approaches. The algorithms for determining each of the features using the NoSQL database are specified. The general scheme of stego-insider detection is also provided. To confirm the efficiency of the approach, an experiment was carried out on a real network. During the experiment, a database of user behavior was collected. Then, user behavior features were retrieved from the database using special SQL queries. The analysis of the results of SQL queries is carried out, and their applicability for determining the attribute is justified. Weak points of the approach and ways to improve them are indicated.

Keywords: cybersecurity; stego-insider; NoSQL database; steganography; attacks

Citation: Kotenko, I.; Krasov, A.; Ushakov I.; Izrailov K. An Approach for Stego-Insider Detection Based on a Hybrid NoSQL Database . *J. Sens. Actuator Netw.* **2021**, *10*, 25. https://doi.org/10.3390/jsan10020025

Academic Editors: Paul Watters, Gregory Epiphaniou, Pedro Pinto, Mohammad Hammoudeh and A. S. M. Kayes

Received: 22 February 2021
Accepted: 25 March 2021
Published: 30 March 2021

1. Introduction

The widespread use of the Internet and modern computer technologies sharply poses the problem of ensuring information security, both at the level of individual organizations and the whole country [1]. One of the ways to compromise security is to create a data transmission channel between the attacker and the attacked system-both for sending malicious commands and programs, and for receiving confidential information.

Typical secure transmission methods based on cryptography turn out to be less effective, since the very fact of encrypting traffic is highly suspicious for any security system. Steganography is considered an alternative approach. Its popularity is largely determined by the fact that it is not subject to the restrictions that most countries in the world impose on the development of their own cryptography methods. The essence of steganography is reduced to the hidden transmission of messages, using typical (i.e.,

constantly used) information objects for embedding data. The condition for such an application is the presence of redundancy in objects, which can be used without violating the structure of the object itself. The effectiveness of this approach is reasonably proved in the work of B. Schneier [2].

Typical examples of the implementation of stego-channels are the following: optional fields in network packets, letters of different encoding with the same spelling in text messages, combinations of machine code instructions with the same effect in executable files, packet delays when transmitting audio traffic. Therefore, for example, using the last feature, you can deliberately delay the sending of some of the packets (which will then be discarded), replacing their contents with hidden information transmitted over this channel. The receiving party, knowing about these features in the transferred objects, can extract secret content from them, which will mean the success of the hidden channel creation.

A separate direction is the hidden embedding of information in the structures of network protocols-network steganography (a term first introduced by G. Simmons in 1983 [3]). The popularity of this direction is explained by the widespread use of the Internet (which expands the scope of application) and the huge volumes of transmitted data (which complicates data analysis). Also, there is many commercial and free programs that allow organizing hidden communication channels on this basis-which an interested attacker will certainly use.

One of the most actual security threats is the theft of commercial information, including the so-called industrial espionage, carried out by violating the security of the organization's perimeter. In recent years, the fight against cybercriminals has shown that internal attacks are becoming increasingly dangerous when an attacker gains access inside the system. In this aspect, the use of steganography methods by internal violators (insiders) (we will call such violators "stego-insiders") significantly increases the success of their attacks, while making it difficult not only to suppress, but also to detect the very fact of an attack [4].

The success of a stego-insider's actions directly depends on the used network environment, and modern technologies (such as cyber-physical systems [5]) can serve as "catalysts" for his actions. In particular, the Wireless Sensor Networks (WSNs) environment (as an Internet of Things applications) turns out to be quite preferable from the point of view of building hidden transmission channels. The following features could be mentioned here: huge scale of distribution (increasing the channel length); heterogeneity of nodes and data (complicating detection); intended for data transmission with a payload (and not, for example, trivial signals or the same type of control commands, embedding into which is difficult); often "sharpened" for the transmission of multimedia content (well suited for stego-embedding); large ranges of deviations in the characteristics of network traffic (which do not allow the use of some approaches to countering attacks by identifying anomalies).

In such an environment, a stego-insider can organize channels in various ways: by generating traffic directly to the network, zombifying (i.e., capturing and controlling) individual network nodes with sensors, adding their own devices with software for stego-embeddings, using sensors in accordance with those laid down in their operating modes. In the latter case, theoretically, a camera can be used that takes images of a static object, minor changes in which will correspond to the transmitted message. Thus, WSNs will be operating normally with a functioning stego-channel. It is also important to note such a feature of WSNs as the large volume, variety and "multi-purpose" of the transmitted information and its use in critical (from the point of view of information security) infrastructure, which obviously increases the interest of violators.

In this regard, the development and study of methods for revealing hidden channels of interaction based on steganography methods created by insiders of organizations, as well as the subsequent provision of the integrity of the controlled environment is an actual task.

The essence of the proposed method for revealing hidden channels is to collect in the network of an organization a set of features of the behavior of its users, concerning their access to the network, applications used, connected devices, organized sessions (continuing the previous author's research [6]). It also takes into account the transfer of

objects suitable for hidden attachments of messages (images, video and audio files, etc.), the use of encryption, the installation of tools for stego-embeddings. A hybrid NoSQL database based on a graph and document-oriented approach is used to store the entire volume of these features. Then, these features are used to determine the features that characterize the stego-insiders. The combination of features with a high degree of probability (and in the long term, using machine learning methods) allows us to distinguish stego-insiders among users.

The main contribution of the paper is as follows:

- The main features of a stego-insider in the network are highlighted, the totality of which makes it possible to determine the fact of the creation of covert transmission channels.
- We created a hybrid NoSQL database based on graph and document-oriented approaches, specialized for storing attributes of stego-insider behavior. This NoSQL database can be deployed for modern large-scale WSNs and the Internet of Things [7,8], and most of the proposed attributes are suitable for collecting information about network activity in these environments.
- Algorithms for determining the features of insiders from the information stored in the database have been developed (partially, using SQL queries), a corresponding experiment has been carried out. The applicability of the stego-insider features for the case of building hidden channels in the WSNs environment has been substantiated and evaluated.
- A hardware-software complex was developed for detecting typical insiders based on their behavior in the network (using methods based on expert rules and machine learning), and an assessment of its quality measures was made.
- Based on the analysis of the obtained experimental results, conclusions are drawn about the prospects for further development of the proposed approach.

The main novelty of the obtained results lies in the created hybrid NoSQL model (as a set of features, their groups and interrelationships), described in an analytical form, as well as in partially formalized features of stego-insiders and algorithms for their determination.

In particular, the novelty in the following.

- The introduced 9 features for the first time reflect the activities of an insider to create hidden data transmission channels;
- The hybrid NoSQL database differs from the existing ones by the ability to store and analyze user characteristics that describe potential insider activity in computer networks, as well as the ability to take into account the dynamics of changes in these characteristics;
- The operation of the algorithms for detecting each of the 9 features is based on the author's elements of the developed NoSQL database model;
- The complex for detecting classic insiders in the network is based on a unique approach to a combination of human and machine analytics: expert rules and machine learning, as well as the use of Big Data;
- Ways of further development of the proposed approach will help to form a "full-fledged trend" of counteraction to stego-insiders that are not detected by existing methods.

The paper is organized as follows. Section 2 provides an overview of relevant works on description models and methods for identifying insiders, as well as methods for embedding and identifying stego-insiders. Section 3 highlights the main features of stego-insiders. In Section 4, a comparative analysis of existing approaches for database construction is made to store the features of user behavior on the network, from which a choice was made to use a hybrid NoSQL database. Section 5 describes algorithms for identifying the features of stego-insiders based on the collected features. Section 6 outlines an experiment for the automated identification of stego-insiders based on a stand integrated into the real network of the organization. Real SQL queries to the database and the received answers

are analyzed. Section 7 provides numerical estimates of the hardware-software complex for classifying users into legal and insiders based on their online behavior. Section 8 discusses the main disadvantages of the proposed solutions and discusses ways to address them.

2. Related Works

In the interests of solving the problem, we will review publications in the following areas, the intersection of which determines the subject area of work: models of insiders and methods for their identification, as well as methods for implementing stego-embedding and their identification. Also, we will separately consider the works devoted to countering insiders in WSNs (since, as noted, this networks have characteristics that simplify the construction of stego-channels).

In Reference [9] it is emphasized that coding psychosocial factors is challenging. In this regard, the paper proposes a description of an organization based on a unified model in the form of a certain automaton, the states of which belong to several classes: safe, reliable, unreliable, and compromised. The behavior of insiders and their threats is investigated using this model.

Reference [10] introduces insider parameters such as predictability, susceptibility, and awareness. The implementation of a computational model based on synthetic scenarios of real insider activity is described. The complexity of evaluating the entered parameters is emphasized.

The solution discussed in Reference [11] aims at identifying deviations in user behavior. The proposed system classifies behavior into two classes-normal and harmful. The algorithms are based on the Artificial Immune System.

In Reference [12] the hypothesis is developed that it is possible to detect insiders not by their behavior, but by the amount of information they excessively collect. In this case, it is not the deviation of users from normal behavior that is considered anomalous, but the deviation in the "data storage profiles" of the users.

Reference [13] discusses a method for embedding text messages into images. To do this, the least significant bit is modified. The algorithm requires a symmetric key exchange between sender and receiver. The main safety problems of stego-images are presented.

Reference [14] provides the results of a study on information hiding in audio files. The goal of the method is to create a less harsh noise ratio and a high-capacity information-hiding scheme (using the discrete wavelet transform).

Reference [15] is devoted to hidden message attachment in TCP/IP packet headers. For this, the generation of a chaotic sequence of the 4th order is used, with the help of which the transmitted message is encrypted. Then, the data is inserted into the identification field of the IP packet header.

The solution proposed in Reference [16] is aimed at detecting stego-embeddings in images. In this case, it is possible to determine not only the fact of embedding, but also the method used. The results also suggest the development of the field of blind steganalysis-when nothing is known about embedding algorithms.

The work [17] is devoted to the problem of detecting hidden channels organized using TCP/IP protocols. In particular, a method is proposed for detecting stego-embedding using initial sequence numbers (ISNs). For the work of the method, it is proposed to use the reconstructed phase space and a statistical model for detecting stego-embeddings.

Reference [18] emphasizes the need to ensure the security of information in WSNs and the severity of internal attacks. The difference between these environments from classical ones is indicated, which requires appropriate modification and adaptation of protection mechanisms. There are three strategies for defending against insider attacks: prevention, detection, and recovery.

Reference [19] also mentions the use of WSNs in critical infrastructures (for example, military or government), which requires their security. The following differences of this network from the classical ones are given, affecting the security mechanisms: limited resources, wireless communication, and possibility of compromising nodes. Mechanisms

for detecting insider attacks based on the facts of data manipulation and compromise are considered.

Reference [20] is devoted to managing secret keys between neighboring nodes in WSNs. A characteristic of the proposed exchange scheme is to counteract internal threats of damage to nodes, and the scheme is based on the location of nodes.

Reference [21] proposes an improvement to the intrusion detection system in WSNs used to defend against internal attacks. The system uses a trust management mechanism in its operation, which requires a reduction in the amount of redundant data from network sensors to improve its efficiency.

Reference [22] describes a security logic scheme for detecting internal attacks in WSNs, which is based on Fuzzy Logic and the classification of nodes into trusted and untrusted. The following are taken as input data for the detection algorithm: physical-signal power, range and residual energy; and network-packet delivery rate, forwarding delay, and transmission rate.

The structure of the overview of relevant works is presented in Table 1.

Table 1. Structure of the overview of related works.

	Insider	**Stego**	**Insider in WSNs**
Attack	[9,10]	[13–15]	[18–22]
Defense	[11,12]	[16,17]	

In conclusion, there is several progressive works regarding the detection of insiders in organizations. The problem of hidden channels based on stego-embedding has also been studied for a long time. However, no works have been found on combining these two topics in the form of counteraction of stego-insiders. Therefore, it is necessary to create new models and methods for detecting stego-insiders.

Analysis of features and variables that are used in models is important for the domain. Each individual model [9–22] used its own features. In our work, we are based on them, but mainly using our own experience.

3. Features of Stego-Insider

In the interests of building a stego-insider model, let us single out the main features by which it differs from legal users.

3.1. Technologies of Stego-Embedding

Hidden channels with the help of stego-embedding can be organized using various technologies. Depending on the technologies used, the following groups of steganography methods are distinguished:

1. Classical steganography – methods of hiding a message in paper documents [23], microfilms and other objects of the material world;
2. Electronic steganography – hiding messages in signals of an analog nature, for example, noise-like carriers, etc. [24,25];
3. Digital steganography – hiding messages in digitized objects that are initially physical or analog in nature (digitized image, text document, audio signal, video signal) [26–28];
4. Computer steganography – hiding data or messages in objects of the program structure of a computer or computer network (in computer programs, in protocols, in sectors of disk space, etc.) [29–32].

The most interesting from the point of view of solving the problem posed are the last two groups of methods. First, it is digital steganography, since the employees of the organization, as part of their job duties, constantly exchange documents, multimedia objects, etc. In addition, second, it is computer steganography, since all the work of the organization's network is based on the exchange of network packets according to standard protocols.

3.2. Features of Digital Steganography

Let us further highlight the features by which it is possible to distinguish the facts of steganographic embeddings of these groups of methods. Let us denote the features using the F_N record (from the word Feature), where N is a serial number.

Steganalysis [33–35] is very popular for detecting hidden messages in individual digitized objects. In addition to the very fact of transmission, it is able (in some cases) to determine the message itself, the methods used, to decrypt it, etc. [36]. There is a whole set of attacks on stego-systems based on a known filled container, known and selected embedded messages, etc. Thus, the use of steganalysis will make it possible to detect the fact of potentially stego-embeddings, which is a feature of insider activity.

Similarly, it is possible to define stego-attachments for WSNs.

Many methods of stego-embeddings are well known and implemented many times, which saves an attacker from having to create their own solutions. Let us make a brief comparative analysis of the main software solutions for stego-embedding messages into digitized objects (Table 2). Empty fields in the table mean the lack of information in open sources.

Thus, the presence of one of these software products on the employee's work computer (together with the peculiarities of his network exchange) may indicate potential insider activity. Please note that just a fact of the presence of such software does not guarantee the activities of an insider in the organization, since it can be used both within the framework of official duties and be accidentally installed. Correlation of this feature with the rest is required.

Determination of this feature in the case of WSNs will have some difficulties, since the software architecture of the network sensors differs significantly from the architecture of Personal Computers or servers. However, hypothetically, with some probability, it is possible to determine the code for stego-embeddings by external demonstration or work logs.

3.3. Features of Computer Steganography

IPv4 is the most widely used protocol on the Internet. Based on its prevalence, it is advisable to use this protocol for the hidden transmission of information. Another advantage of using this protocol is the high resource consumption of steganalysis of such stego-containers. Presumably, to identify steganographic channels, a sample of intercepted connections, in general, and individual network packets, in particular, as well as a statistical analysis of the data obtained, is required.

For hidden transmission of information in an IPv4 packet, you can use the "Identification" field (16 bits), and to mark the sent packets, use the "Differentiated Services Code Point" field (6 bits). The value of the "Identification" field of an IP packet is generated on the sender side and contains a random number that is generated during packet creation. The "Identification" field only applies when fragmentation is used. Therefore, to use this method, it is necessary to know the MTU value in the transmitted network and not exceed it, so that the packet is not fragmented during transmission. If there is no need for packet fragmentation, a certain redundancy also appears in the "Flags" field, in the second bit, which is responsible for setting the "Don't Fragment" (DF) flag. It is possible to specify a flag that notifies the sender that the sender does not want to fragment the packet. If the packet with the steganogram will not be fragmented due to its size, you can hide the information in the "Do Not Fragment Bit" flag field. Using this method provides 1 bit bandwidth. The Differentiated Services Code Point is used for traffic classification and traffic management. Depending on the value of this field, some packets will be processed first, will not be dropped, or vice versa will be dropped before others (which is alternative to SDN). The use of this field is possible if there is no channel congestion along the path of the packets. Otherwise, you will have to send the failed packets again.

Table 2. Comparative table of the main software solutions for stego-message embedding.

Software	Input Data	Output Data	Encryption
QuickStego	Text, graphic	BMP, JPG, GIF, WAV, MP3	Exist
Steganos Privacy Suite	Graphic, audio, video	Graphic, audio, video	AES-XEX
Xiao Steganography	Text, graphic	BMP, WAV	RC4, DES etc.
OpenStego	Text, graphic	PNG	Password
SilentEye	Text, graphic	BMP, WAV, JPG	AES128, AES256
SSuite Picsel Security	Text, graphic	BMP, WAV, JPG, WMF	-
StegoStick	Text, graphic, audio, video	Graphic, audio, video	DES, Triple DES, RSA
DeepSound	FLAC, APE, WAV, MP3	FLAC, APE, WAV, MP3	AES-256
Trojan	JPEG, BMP, GIF, PNG, PCS etc.	BMP, PNG, TIF	Password
Camouflage	-	-	-
OpenPuff	Images, audio, video, flash etc.	Images, audio, video, flash etc.	Using key phrases
Steghide	JPEG, BMP, WAV	JPEG, BMP, WAV	Blowfish, MD5
mp3stego	MP3	MP3	Password
SteganPEG	JPEG	JPEG	Password
SteganographX Plus	BMP	BMP	-
Crypture	BMP	BMP	Password
Portable SteganoG	-	-	-
rSteg	TCP packets	TCP packets	-
Visual Steganographic Laboratory	JPEG	JPEG	-
SteGUI	JPEG, BMP, WAV	JPEG, BMP, WAV	Blowfish, MD5
stegano	PNG	PNG	-
cloackedpixel	PNG, JPG	PNG, JPG	Exist
LSBSteg	PNG, BMP	PNG, BMP	-
Spam mimic	Text	Text	Password
Contraband	Any	BMP 24 bit	-
FFEncode	Text	Text	External
DANTSOVA	-	-	-
SecurEngine Professional 1.0	Text	Text	AES, Gost, BlowFish, ThreeDes
ImageSpyer G2	BMP, JPEG, WMF, EMF, TIFF	BMP, JPEG, WMF, EMF, TIFF	30 encryption and 25 hash algorithms
F5	JPEG	JPEG	-
StegoStick (beta)	BMP, JPG, GIF, MPG, WAV etc.	jBMP, JPG, GIF, MPG, WAV etc.	-
Secure Secret	-	-	-
Nicetext	Text	Text	-
StegParty	-	-	-
Markov-Chain-Based	-	-	-
Gif-It-Up 1.0	GIF	GIF	-
EZStego	GIF, PICT	GIF, PICT	-
DiSi-Steganograph	256-color PCX	PCX	-
Hide and Seek	DIB, BMP, VOC, WAV etc.	DIB, BMP, VOC, WAV etc.	BlowFish
Steganos	Text, graphic	BMP, DIB, VOC, WAV, ASCII, HTML	-
Jsteg	JPEG	JPEG	-
DeEgger Embedder	BMP, JPEG, PNG	AVI, JPG, PNG, MP3 etc.	-

The bandwidth of such a channel will be calculated as follows:

$$b_i = 16 \times \frac{a - (c_{IPv6} \times s_{IPv6})}{s_{IPv4}}, \quad (1)$$

where b_i – maximum throughput of a steganographic channel build on the basis of the "Identification" field, maximum throughput of a steganographic channel built on the basis of the "Identification" field, a – maximum channel bandwidth, c_{IPv6} – the number of

IPv6 packets transmitted per second, s_{IPv6} – average IPv6 packet size transmitted on a channel, s_{IPv4} – IPv4 packet size usable for steganographic embedding. This formula is correct, provided that all outgoing IPv4 traffic will be sent to the recipient of the network steganogram, and confirmation of receipt of packets is not required.

In the case of WSNs, the feature definition will be similar.

Thus, the suspicious content of the specified fields of the IP packet and the chronology of their changes may be a feature of insider activity.

In the TCP protocol, the "source port" field is suitable for hidden information transfer, the size of which is 16 bits. The TCP header contains the acknowledgment number (32 bits) and sequence number fields, so it makes no sense to use the "Differentiated Services Code Point", "Explicit Congestion Notification" fields in the IPv4 header to mark the packet. Therefore, the freed space of 6 bits can be used to transfer different files at the same time.

The bandwidth of such a steganographic channel is calculated as follows:

$$b_s = (s_{sp} + s_{an}) \times \frac{a - (c_{IPv6} \times s_{IPv6}) - (c_{OIPv4} \times s_{OIPv4})}{s_{TCPIPv4}}, \quad (2)$$

where b_S – maximum throughput of a steganographic channel built based on the "source port" and "acknowledgment number" fields, s_{sp} – size of source port (16 bits), s_{an} – size of acknowledgment number (32 bits), a – maximum channel bandwidth, c_{IPv6}–the number of IPv6 packets transmitted per second, s_{IPv6} – average IPv6 packet size transmitted on a link, $s_{TCPIPv4}$ – the size of an IPv4 packet that contains TCP blocks, c_{OIPv4} – the average number of packets not containing TCP blocks, s_{OIPv4} – the average size of packets without TCP blocks.

3.4. Behavioral Features

The previously indicated features can work when analyzing individual entities-transmitted objects, network packets, sessions. Nevertheless, it is possible to provide behavioral features that allow one to take into account the features of the sets of these objects and their internal relationships. This will allow for deeper detection of trained stego-insiders. The stego-insiders, obviously, will try to hide or cover up the traces of their malicious activity in terms of the channels being created.

In the case of WSNs, the feature definition will be similar.

It is obvious that the creation of a hidden channel by an attacker is necessary for organizing constant data exchange, and not for sending a message once. Thus, it can be assumed that in the case of digital steganography, an attacker will exchange many images to send a message, receive a response, possibly confirm receipt, etc. Thus, such specific behavioral activity is a feature of insider activity.

As with the previous one, the exchange of audio, video and text recordings can be used as additional features.

It is important to note that apart from others, these features do not guarantee the identification of a stego-insider, since many employees (for example, designers) carry out such activity in the course of their work, while being loyal to the company.

In the case of WSNs, the feature will mention of a stego-insider with less confidence, since this environment is just intended to transfer groups of objects of the same type (including well suited for stego-embeddings), which will introduce a type I error feature into the detection results.

When creating a hidden messaging channel, it is highly likely that an attacker will use one or a few methods of stego-embedding. Thus, it can be assumed that the size and basic content of the stego-container will be similar in the case of multiple exchanges of messages. Finally, small differences in the objects transmitted over the network can be attributed to features of insider activity.

As with the F_6 feature, the reliability of the stego-insider detection in the WSNs environment will be reduced. Therefore, for example, if a network node is a device that

sends images of a monitored object with some frequency, then most of these images will most likely be almost identical-which is quite natural.

Insider activity mainly involves theft of confidential information to transfer it outside the perimeter of the organization. Stego-insiding imposes on this limitation in the form of the use of hidden transmission channels. Thus, the transfer of the stolen information should be carried out to some external IP address (or a set of it, to complicate detection). Hence, it is also a feature of insider activity.

In most cases, WSNs should not access nodes outside the perimeter of the organization, and therefore, the feature may correspond to attempts to transfer information to third parties.

3.5. Antidetection Features

Insiders, organizing stego-channels, are obviously better prepared than simple not loyal employees who steal information using USB drives or send archives with documents to a personal mailing address. Thus, such stego-insiders will be more prepared (qualified) and tend to hide their activities. It can also be used as features of both the very fact of insider activity, as well as to determine the level of its preparedness.

The presence of encrypted data, especially sent outside the perimeter of the organization, is regarded by many security systems (DLP, etc.) as a fact of confidential information leakage. Therefore, trained attackers will strive to avoid using encryption. This is a feature of the preparedness of the offender.

The behavior of an attacker in the WSNs environment will be determined by the same features.

Similar to the previous feature, an insider is more likely to behave like any legal employee. He will minimize all his deviations from the accepted norms – come to work on time, log in only from approved devices, be in a psychosocial stable state, enter the password from the first attempt (to avoid suspicion of brute-force and account hacking). Consequently, in the case of stego-insiders, their suspiciously "correct" behavior will just say the opposite-about their "abnormality".

As mentioned earlier, WSNs have greater heterogeneity in the characteristics of network traffic, which indicates a decrease in the significance of this feature.

All these features can be determined by analyzing the static and dynamic characteristics of users and the devices they use. However, due to the huge amount of information collected, it is necessary to store it in a single integrated database, suitable for operational data collection and processing. In the interests of this, a database model (DB) was created and implemented, a description of which is given below.

4. Stego-Insider Model

Modern information systems require a special organization of work with the incoming data, appropriate processing and recording of a certain type of data in the database. Database management systems (DBMS) are high-level software for the distribution, processing, storage and presentation of data, working with low-level application programming interfaces (API). To solve various problems, different types of DBMS and their implementations were created, which are based on various database models. Any DBMS are created to work with one of them, taking into account the features of information processing and implements one of the database models for the logical separation of data. These models define how information will be processed and how information is managed in an application.

The problem is that not all database management models are sufficiently adapted to the timely processing of large volumes of information and events. The specificity of cybersecurity problems lies in the need to use new database models and the use of big data technologies for processing computer network traffic. The approach proposed below differs from the existing ones using big data technologies based on the combined use of document-oriented and graph database models. This approach is aimed at increasing the efficiency of processing large volumes of traffic.

4.1. Prerequisites for Creation

Consider various database models for an informed choice of the most suitable one to solve the current research problem. Let us select the necessary properties that the database model must correspond for processing traffic: consistency-the property implies the preservation of data integrity after performing any operations; availability-the property of the system to run smoothly; separation resistance is a property that characterizes the performance of the database even if communication between individual cluster nodes is broken.

In theory, it is considered impossible to achieve the fulfillment of all these three mutually contradictory requirements. Standard SQL databases are an unallocated cluster in which all node elements are interconnected, and if the connectivity is broken, the performance of the database is also disrupted.

Due to the distribution and the need to process large volumes of not always structured data, as well as their diversity, SQL databases cannot cope with their task. In this case, it is relevant to use the NoSQL database model [37], which ensures data consistency at the expense of their availability. A NoSQL database model should be understood as a way of structuring data, which consists of getting rid of restrictions on storing and processing information. NoSQL databases take an unstructured approach to organizing data, but offer efficient ways of processing data. NoSQL DBMSs do not use a relational model, but allow specific solutions with the possibility of unlimited formation of records and storage of data in the form of key-value, which can be represented as a storage of column families. They also allow you to group collections of data with other databases by aggregating and storing them as a whole. Such databases can represent a single object and at the same time correctly respond to queries to fields.

Let us make a formal comparison of SQL and NoSQL databases (Table 3), and also indicate the types of data that should be taken into account when processing computer network traffic (Table 4).

Table 3. Comparison of SQL and NoSQL.

SQL Databases	NoSQL Databases
Relational and non-distributed	Non-relational and distributed
Based on tables and relationships between them	Can be document-oriented, key-value, graph, column
Store data in strings	Store data in collections of values
Have predefined schemes	Have dynamic schemes
Vertically scalable	Horizontally scalable
Work with structured data	Can work with any data type

Table 4. Basic data types.

Types	Description
Structured	Stored in a fixed field within a record
Unstructured	Content is context sensitive and variable
Machine	Information is automatically generated by the computer
Graph	Data is presented as a graph
Stream	Enter the system when events occur, rather than being loaded into storage in large arrays
Batch	Specifically formed blocks of data transmitted over the network in packet mode

The approach to using NoSQL databases has arisen as a result of the fact that classical relational databases are ill-suited to the huge and rapidly growing volume of information that needs to be processed by computer systems, which presents a serious engineering challenge. However, a way out of this situation was found by creating databases that run on several machines in a distributed environment and do not depend much on the type of information being processed.

There are several types of basic NoSQL databases, depending on the task being performed [38]: key-value, column, document-oriented, and graph.

The data model is represented as a collection of key-value pairs, but not combined into documents. It provides fast data lookups, but lacks a clear schema, providing a distributed hash table that allows data to be written to a specific key and read back using that key. Such databases are easily scalable and have the lowest search latency.

In column database individual data is stored in cells grouped into columns that are logically structured into families and can consist of an almost unlimited number of columns created during program execution or schema definition. Reading and writing is done using columns, not rows. They also differ from the usual ones, for each column from the table a separate file is created containing data from the column. This structure allows you to aggregate data and execute certain queries more efficiently than in an ordinary database.

The data model in document-oriented databases is a collection of key-value connections, combined into documents. They provide the ability to store complex nested documents and can be used, for example, when storing information about a user with several addresses in the system. Such a base allows you to store a complex object and frees you from the need to create several tables, combining them into one. Thus, it is not necessary to have a scheme, which provides additional flexibility and simplifies modifications.

The data model in graph databases is represented as a parameterized graph. The graph consists of anchor points and edges, with the anchor points acting as objects and the edges acting as links between objects. The graph also consists of properties associated with anchor points. It uses an approach called "no-index adjacency", meaning that each anchor point includes a pointer to an adjacent anchor point. With this approach, you can work with a large number of records. In the database, the main emphasis is on the connection between data, providing schema-free storage of semi-structured data. Graph databases provide all graph operations such as search in length, width, shortest path search, but such databases are difficult to cluster.

The above descriptions of various types of databases suggest that to solve the problem of processing traffic of computer networks, it is advisable to use both a document-oriented database model for collecting data with subsequent information aggregation, and a graph model to facilitate the interpretation of the results. In this regard, a hybrid approach can be applied to solve this problem, which implies the use of a combination of several database models, which should lead to an increase in the efficiency of processing information obtained from traffic.

Let us analyze research papers using the NoSQL model.

Reference [38] notes the need to use NoSQL databases due to the increasing demand to store and process huge amounts of data. It compares the four NoSQL database models listed above in terms of performance, scalability, flexibility, complexity, and functionality. Given the growing need for managing big data and unstructured business transactions, it is argued that graph NoSQL databases are well suited for data that has complex relationship structures, while at the same time ease of use is achieved through the use of key-value stores.

Reference [39] compares common NoSQL database models based on application, strengths, and limitations. The authors conclude that graph databases are fast in processing with support for querying related datasets and are suitable for displaying entity-relationship type diagrams. A limitation of graph databases is the lack of a standardized API or query language for accessing databases. In addition, splitting large graphs decreases system performance due to the large number of relationships between graph vertices.

Reference [40] provides a comparative analysis of relational and NoSQL databases. It is noted that relational databases are based on the ACID (Atomicity, Consistency, Isolation, Resilience) model, which provides better consistency and security, and uses the standard query language (SQL). However, relational models are less scalable, lose in performance, can process a limited amount of data, and have problems in ensuring availability when the number of database users increases. NoSQL databases are based on the BASE model (basic availability, volatile state, consistency), which offers great scalability, better performance

when using a significant amount of data; however, they have security problems and using a standardized structured query language.

Reference [41] notes that with the rapid growth of traffic and the development of cloud computing, there is a need to create a DBaaS (database as a service) model. DBaaS is a data management paradigm where a third-party service provider stores a database and provides the appropriate software and physical infrastructure to maintain it. DBaaS supports both SQL and NoSQL databases. DBaaS is argued to be suitable for big data processing.

In Reference [42], it is noted that in any distributed computing implementation, it is possible to provide no more than two of the following three properties: consistency, availability, and partition tolerance. In particular, it was concluded that relational databases are not partition resistant. The non-relational databases MangoDB and HBASE have consistency and partition resistance properties, while CouchDB and DynamoBD have accessibility and partition resistance properties.

Reference [43] describes an integrated storage system (using Protégé 5.1.0) based on two approaches: relational and ontological. The main application of the model is in the storage and processing of data related to information security. The system has the flexibility of internal data representation and the ability to make logical conclusions. It is assumed that the solution will be in demand in the field of intelligent information security systems.

4.2. Hybrid Model

An analysis of relevant works related to solving various problems based on NoSQL models allows us to conclude that the use of a hybrid model is an adequate approach to solving the problem of traffic analysis.

To extract useful information from the analyzed traffic for the analysis of anomalies, it should be taken into account that the analyzed network packets contain many records that are conveniently represented in a graph model in combination with a document-oriented model. This hybrid approach to storing tables in a database can reduce the amount of data that needs to be loaded from disk. In the document-oriented model, collections of elements are transparent to the database, which allows you to query their fragments and perform partial data retrieval.

For the practical implementation of this hybrid approach, it is proposed to use a NoSQL DBMS, which can combine the capabilities of both document and graph data models. An example of such a DBMS is OrientDB, which combines the convenience of a graph model, establishing dependencies between objects using parameterized links, and the flexibility of the document model, allowing you to store many complex records.

The requirements for a hybrid model can be attributed to its problem-orientedness, since it must store a system of stego-insider features. Also, the model should be dynamic – built as user activities in the organization are defined. Based on this, the key objects of the model and their connections can be selected.

The features of the model in terms of detecting stego-insiders include the following:

- store information about the fields of IP and TCP network packets, since they can detect hidden transmission channels;
- collect information about the use of software tools to identify among them the software for stego-embedding (shown in Table 1);
- contain the minimum necessary information (to reduce the size of the database) about objects sent in the network; this will allow both to apply steganalysis to them and to determine behavioral features;
- display all connections to external nodes to organize;
- to deeply detect trained insiders, their suspicious “normality” should be revealed by the facts of the absence of encrypted channels and proximity to the average behavior of legal employees.

All these features should be identified by a single model representation of data collected from the network (including the organization’s devices-routers, personal computers, etc.).

4.3. Model Objects

The concept of classes in OrientDB is based on an object-oriented programming methodology, and classes are data types that follow certain rules.

Classes can be equated with tables that contain fields, have characteristics, can inherit from superclasses, and be abstract. A distinctive feature of OrientDB is that it is not necessary to specify the type and list of fields of a record when creating a class or the record itself, and the set of fields can be individual for each record. For a graph model, the class must inherit from the corresponding superclass – Vertex (vertex) or Edge (edge).

All objects in the nodes of the model can be attributed to one of the classes: superclass, which is the main one in the hierarchy; and an inherited class – which extends the properties of the superclass. Vertex is an entity represented as a superclass that can be connected to other vertices and has the required properties: a unique identifier, a set of incoming and outgoing edges.

Edge is an entity that is a superclass that connects two vertices and has the required properties: an identifier, a link to an incoming and outgoing vertex, a label to determine the type of relationship between vertices. Relationships unite documents using parameterized links between classes, using the persistent pointer mechanism between records, which ensures high speed.

The User class inherits from the Vertex superclass and stores unique data about user login actions.

The Application class inherits from the Vertex superclass and stores information about the installed software on the device.

The Device class inherits from the Vertex superclass and stores information about connected and networked devices.

The Session class inherits from the Vertex superclass and stores information about a specific network session. In particular, the class contains information regarding Devices, Applications bound to it, transmitted files (of course, text, images, audio and video are of the greatest interest) and low-level information about network packets.

The Relation class inherits from the Edge superclass and stores information about the type of relationship between objects with the Edge superclass. Based on the given inherited classes, the possible types of relationships between it are shown in Table 5.

Table 5. Types of links between objects.

Type	Initial Vertex	End Vertex
DeviceToSession	Device	Session
AppToSession	Application	Session
SessionToUser	Session	User

The database operates on records as strings. A row always contains a METADATA column including the following fields: RecordID (@rid) – row identifier, Class (@class) belonging to a class, record version (@version) – version in the table. The main record type is a document belonging to a specific class. It specifies its type, and it has a set of features called PROPERTIES.

Fetching allows you to get a graph from documents using a query language based on SQL. The features discussed above allow you to flexibly work with data, the set of parameters of which is initially unknown or may change during operation, which is convenient for processing traffic. Thus, OrientDB is an acceptable choice for traffic processing tasks.

At the same time, a hybrid model should have such features as the absence of the need to specify a data schema, visibility of presentation, scalability, the ability to specify complex data, the ability to specify data with many links.

4.4. Model Structure

To build a model, it is necessary to collect data of the following categories: static – on the composition of devices in the organization's network, as well as dynamic – on

user behavior in the organization's network. At the same time, some of the data will be attributed to both of these categories. Therefore, for example, the user's connected mobile device has both a MAC address (first category) and a time of connection to the network (second category).

The following scanning points were selected as sources for collecting the information required for the model: Packet – transmitted network packets, User – information about basic user actions, Session – collected network sessions, Device – connected devices, Object – objects transmitted over the network (audio-video files etc.), Application – applications running on devices. The implementation of scanners from each of the sources allowed us to create a solution that collects the following user behavior features [44–46].

For Packet scanners:

- PacketIPv4Identifier – identifier field (16 bits);
- PacketIPv4DSCP – differentiated services code point (6 bits);
- PacketIPv4DNFBit – bit of the flags field, which is responsible for the sender's unwillingness to fragment the packet (1 bit);
- PacketTcpSrcPort – packet source port (32 bits).

For User scanners:

- UserID – unique user number;
- UserLogin – user login;
- UserPass – user password;
- UserTotalAuth – number of user login attempts to the system;
- Sessions – list of sessions used by the user.

For Session scanners:

- SessionID – unique session number;
- SessionData – information about a specific unique session;
- SessionSites – a list of URLs (internal and external) with which the session was connected;
- SessionTime – time from the beginning of the session;
- SessionLoginType – method of logging into the system (from the workplace, through a virtual private network, etc.);
- SessionCrypto – fact of encryption in the session (optional, its type).
- DeviceID – device creating the session;
- ApplicationID – application creating the session;
- Objects – objects transferred during the session;
- Packets – low-level information about network packets (no full content).

For Device scanners:

- DeviceID – unique device number;
- DeviceName – device name;
- DeviceOSType – name of the operating system;
- DeviceGeo – geographic location (abrupt change in geographic location when working in the system can be considered suspicious);
- DeviceVendor – manufacturer's name;
- DevicePeriph – used peripheral devices.
- Applications – applications installed on the user's device

For Object scanners:

- ObjectType – type of the object transmitted over the network;
- ObjectHash – hash of the content of the object transmitted over the network;
- ObjectCrypto – fact that the object is encrypted (optional, its type).
- ObjectStegoAnalysis – result of applying steganalysis to the content of an object.

For Application scanners:

- ApplicationID – unique application number;
- ApplicationName – software name;

- ApplicationVersion – software version;
- ApplicationDeveloper – software developer;
- ApplicationSrcPort – source ports used by the application;
- ApplicationDstPort – destination ports used by the application.

The features collected in this way, then stored in the database, can be used to highlight the features of stego-insider activity. To save the size of the database, instead of storing the entire contents of the transferred objects, their hash (ObjectHash feature) is used instead.

The analytical form of the model can be written as a union of components as follows:

$$Model = Packets \cup Users \cup Sessions \cup Devices \cup Objects \cup Applications. \quad (3)$$

Each of the components contains the following elements:

$$\begin{cases} \text{Packets} = \bigcup \left\langle \begin{array}{c} PacketIPv4Identifier, PacketIPv4DSCP, \\ PacketIPv4DNFBit, PacketTcpSrcPort \end{array} \right\rangle \\ \text{Users} = \bigcup \left\langle \begin{array}{c} UserID, UserLogin, \\ UserPass, UserTotalAuth, Sessions \end{array} \right\rangle \\ \text{Sessions} = \bigcup \left\langle \begin{array}{c} SessionID, SessionData, \\ SessionSites, SessionTime, \\ SessionLoginType, SessionCrypto, \\ DeviceID, ApplicationID, Objects, Packets \end{array} \right\rangle \\ \text{Devices} = \bigcup \left\langle \begin{array}{c} DeviceID, DeviceName, \\ DeviceOSType, DeviceGeo, \\ DeviceVendor, DevicePeriph, \\ Applications \end{array} \right\rangle \\ \text{Objects} = \bigcup \left\langle \begin{array}{c} ObjectType, ObjectHash, \\ ObjectCrypto, ObjectStegoAnalysis \end{array} \right\rangle \\ \text{Applications} = \bigcup \left\langle \begin{array}{c} ApplicationID, ApplicationName, \\ ApplicationVersion, ApplicationDeveloper, \\ ApplicationSrcPort, ApplicationDstPort \end{array} \right\rangle \end{cases}. \quad (4)$$

Thus, the components of the model have the following relationships. Each User has an associated list of Sessions. Each Session is associated with its own device (defined by DeviceID), which creates the Application session (defined by ApplicationID), transmitted in Objects sessions (each of which is identified by the ObjectHash hash), used when transmitting the features of network packets Packets. Also, a list of applications installed on it is associated with each device.

The generalized graphical diagram of the model is shown in Figure 1.

According to the scheme (Figure 1), the construction of the database graph is as follows. When a new Device is connected, an object of the corresponding class is created. Then, when creating a Session, an object corresponding to the network data transmission channel is bound to the Device. Likewise, a class object is created and bound to the session when the web application is launched. Session is bound to User, for which the corresponding object is created. If the object of the class is already present on the graph, then its duplicate is not created, but only the binding to the existing one is carried out.

The structure of the database (including the above features) can be represented as the following diagram.

The blue background in Figure 2 denotes the fields that refer to others, and the green–the fields that contain arrays of references to others.

The implementation of the presented database model was implemented using the OrientDB DBMS. This product provides a full-fledged Web-interface for working with the database (including SQL-requests), for example, viewing information on a specific entity (Figure 3).

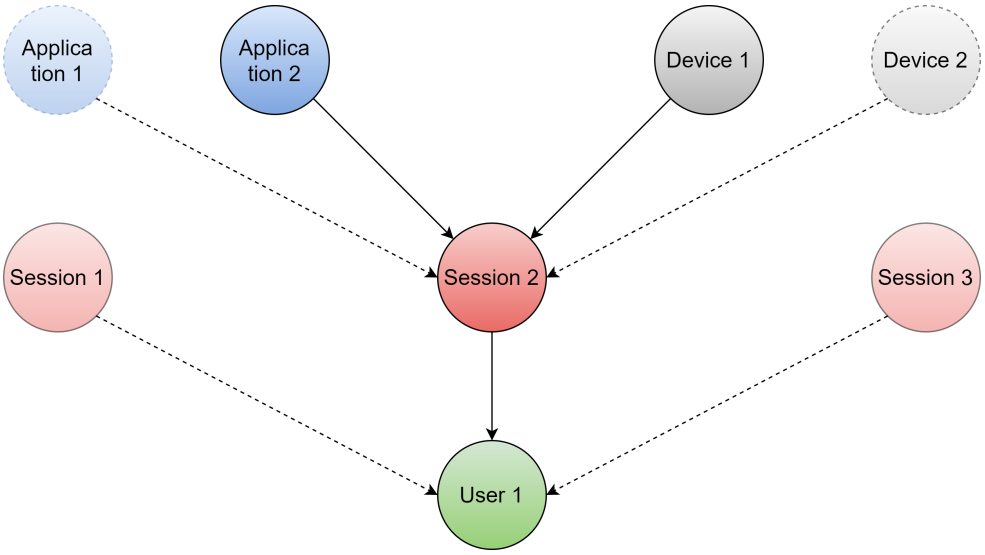

Figure 1. Generalized graphical diagram of the model.

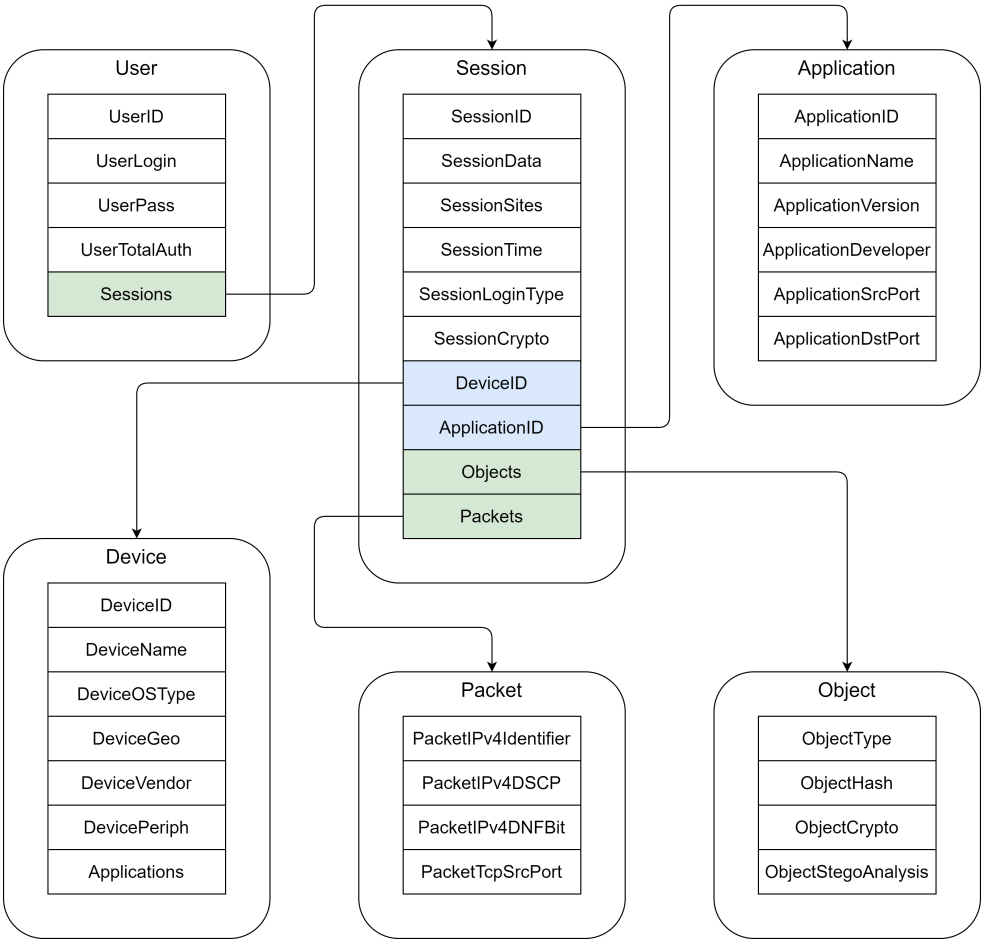

Figure 2. Scheme of database with features.

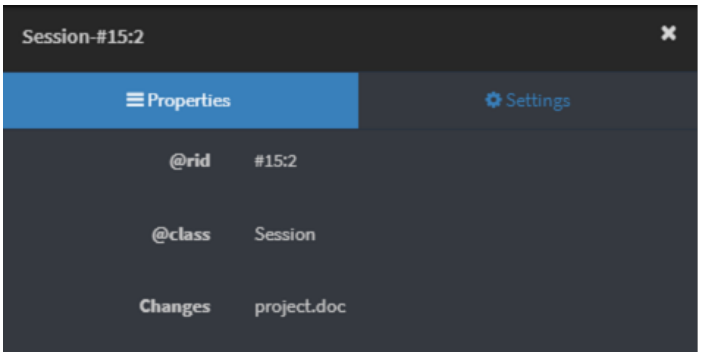

Figure 3. Example of partially information about Session entity.

5. Stego-Insider Detection

As mentioned earlier, the features stored in the database model are used to highlight the stego-insider features. The generalized algorithm for detecting an insider in the form of a pseudocode can be described as Algorithm 1.

The algorithm accepts a hybrid NoSQL database as input, containing the collected features of user behavior on the network. The output of the algorithm is a list of certain insiders. The principle of operation of the algorithm is based on the collection of features based on the features stored in the database. For this, a tuple of features is determined for each user, which are then classified according to the user's relationship to the class of insiders. Below are the search algorithms for each of the F_N features.

Algorithm 1: InsidersDetector.

```
   Input:
   DataBase – NoSQL DataBase with user's features
   Output:
   Insiders – List of detected insiders
 1 begin
 2     List<User> Insiders;
 3     Users = DataBase.GetUsers();
 4     foreach User in Users do
 5         Array<Feature> Features;
 6         Features[1] = CheckFeature_1(DataBase, User);
 7         Features[2] = CheckFeature_2(DataBase, User);
 8         Features[3] = CheckFeature_3(DataBase, User);
 9         Features[4] = CheckFeature_4(DataBase, User);
10         Features[5] = CheckFeature_5(DataBase, User);
11         Features[6] = CheckFeature_6(DataBase, User);
12         Features[7] = CheckFeature_7(DataBase, User);
13         Features[8] = CheckFeature_8(DataBase, User);
14         Features[9] = CheckFeature_9(DataBase, User);
15         Class = PerformClassification(Features);
16         if Class == Class.Insider then
17             Insiders.Add(User);
18         end
19     end
20     return Insiders;
21 end
```

5.1. Algorithm for F_1

The feature of the presence of stego-embeddings is identical to the ObjectStegoAnalysis feature of Object. The generalized algorithm for checking this feature in the form of pseudocode can be described as Algorithm 2.

The algorithm accepts a hybrid NoSQL database and a verified user as input. The output of the algorithm is the fact of confirming the feature of the insider. The principle of the algorithm is based on the following steps. First, all sessions of a given user are searched. Secondly, all objects transmitted within the session are searched. In addition, thirdly, each object is checked for the potential presence of stego-embeddings in it (according to the result of the earlier steganalysis algorithm). If found, the algorithm returns confirmation of the stego-insider feature, otherwise – a refutation.

Algorithm 2: Algorithm for Feature 1.

```
Input:
DataBase – NoSQL DataBase with user's features
User – User for check the insider's feature
Output:
Flag – Boolean flag of insider's feature
1  begin
2      Flag = False;
3      Sessions = DataBase.FindSessionsByUser(User);
4      foreach Session in Sessions do
5          Objects = DataBase.FindObjectsBySession(Session);
6          foreach Object in Objects do
7              if Object.ObjectStegoAnalysis == True then
8                  Flag = True;
9              end
10         end
11     end
12     return Flag;
13 end
```

5.2. Algorithm for F_2

The feature of the software installed on the user's device can be determined by the combination of ApplicationName, ApplicationVersion, and ApplicationDeveloper features of Application. Also, more complex algorithms for identifying such software by instruction templates [47,48] are used. The rest of the Application features are auxiliary and can provide additional information to the expert (Algorithm 3).

The algorithm accepts a hybrid NoSQL database and a verified user as input. The output of the algorithm is the fact of confirming the feature of the insider. The principle of the algorithm is based on the following steps. First, all sessions of a given user are searched. Secondly, all devices that initiated these sessions are searched for. Thirdly, all applications installed on these devices are searched. In addition, fourthly, each application is checked for the presence in the database of software for stego-embeddings. If found, the algorithm returns confirmation of the stego-insider feature, otherwise—a refutation.

5.3. Algorithm for F_3

The feature of stego-embedding into the fields of an IP packet can be determined by the PacketIPv4Identifier, PacketIPv4DSCP, PacketIPv4DNFBit features of the Packet. In this case, you will need to obtain a sequence of data collected from these fields, and then apply steganalysis to them (Algorithm 4).

The algorithm accepts a hybrid NoSQL database and a verified user as input. The output of the algorithm is the fact of confirming the feature of the insider. The principle of the algorithm is based on the following steps. First, all sessions of a given user are searched. Second, all IP packets, transmitted within the session, are searched (more precisely, the minimum information required about their headers). Third, the header fields used for stego-embedding are selected from the packets. In addition, fourthly, the data in these fields

is checked for the potential presence of stego-embeddings. In the case of a positive check, the algorithm returns confirmation of the stego-insider feature, otherwise – a refutation.

Algorithm 3: Algorithm for Feature 2.

```
Input:
DataBase – NoSQL DataBase with user's features
User – User for check the insider's feature
Output:
Flag – Boolean flag of insider's feature
1  begin
2      Flag = False;
3      Sessions = DataBase.FindSessionsByUser(User);
4      foreach Session in Sessions do
5          Devices = DataBase.FindDevicesBySession(Session);
6          foreach Device in Devices do
7              Applications = DataBase.FindApplicationsByDevice(Device);
8              foreach Application in Applications do
9                  if StegoSoftwareBase.IsContains(Application) == True then
10                     Flag = True;
11                 end
12             end
13         end
14     end
15     return Flag;
16 end
```

Algorithm 4: Algorithm for Feature 3.

```
Input:
DataBase – NoSQL DataBase with user's features
User – User for check the insider's feature
Output:
Flag – Boolean flag of insider's feature
1  begin
2      Flag = False;
3      Sessions = DataBase.FindSessionsByUser(User);
4      foreach Session in Sessions do
5          Packets = DataBase.FindPacketsBySession(Session);
6          foreach Packet in Packets do
7              F1 = Packet.PacketIPv4Identifier;
8              F2 = Packet.PacketIPv4DSCP;
9              F3 = Packet.PacketIPv4DNFBit;
10             if StegoAnalyser.CheckIPv4Fields(F1, F2, F3) == True then
11                 Flag = True;
12             end
13         end
14     end
15     return Flag;
16 end
```

5.4. Algorithm for F_4

The feature of stego-embedding in the TCP packet fields is defined similarly to F_3, but by the PacketTcpSrcPort feature (Algorithm 5).

The algorithm accepts a hybrid NoSQL database and a verified user as input. The output of the algorithm is the fact of confirming the feature of the insider. The principle of the

Algorithm 5: Algorithm for Feature 4.

```
Input:
DataBase – NoSQL DataBase with user's features
User – User for check the insider's feature
Output:
Flag – Boolean flag of insider's feature
1  begin
2      Flag = False;
3      Sessions = DataBase.FindSessionsByUser(User);
4      foreach Session in Sessions do
5          Packets = DataBase.FindPacketsBySession(Session);
6          foreach Packet in Packets do
7              F1 = Packet.PacketTcpSrcPort;
8              if StegoAnalyser.CheckTCPFields(F1) == True then
9                  Flag = True;
10             end
11         end
12     end
13     return Flag;
14 end
```

algorithm is based on the following steps. First, all sessions of a given user are searched. Secondly, all TCP packets (more precisely, the minimum information required about their headers) transmitted within the session are searched. Third, the header field used for stego-embedding is selected from the packets. In addition, fourthly, the data in this field is checked for the potential presence of stego-embeddings. In the case of a positive check, the algorithm returns confirmation of the stego-insider feature, otherwise – a refutation.

5.5. Algorithm for F_5

The feature of the presence of many the same type of images, text, audio, video files can be determined by the number of objects of the same type associated with the Session – the Objects feature (Algorithm 6).

The algorithm accepts a hybrid NoSQL database and a verified user as input. The output of the algorithm is the fact of confirming the feature of the insider. The principle of the algorithm is based on the following steps. First, all sessions of a given user are searched. Secondly, all objects transmitted within the session are searched. In addition, third, the number of objects of each type is calculated. If the limit for a certain type is exceeded, the algorithm returns confirmation of the stego-insider feature, otherwise – a refutation.

5.6. Algorithm for F_6

The feature of a small difference in the transmitted images, text, audio, video files can be determined as follows. First, objects of the same type are selected by the ObjectType feature of Object. Then, they are compared using FuzzyHash [49,50] using the ObjectHash feature. The idea of such a hash is that it allows you to compare objects with similar content (Algorithm 7).

The algorithm accepts a hybrid NoSQL database and a verified user as input. The output of the algorithm is the fact of confirming the feature of the insider. The principle of the algorithm is based on the following steps. First, all sessions of a given user are searched. Second, all objects transmitted within the session are searched. Third, groups of the same type are selected from these objects. In addition, fourth, objects of the same type are checked for similarity using FuzzyHash. In case of similarity, the algorithm returns confirmation of the stego-insider feature, otherwise – a refutation.

Algorithm 6: Algorithm for Feature 5.

```
Input:
DataBase – NoSQL DataBase with user's features
User – User for check the insider's feature
Output:
Flag – Boolean flag of insider's feature
1  begin
2      Hash<Type, int> TypesCounter;
3      Flag = False;
4      Sessions = DataBase.FindSessionsByUser(User);
5      foreach Session in Sessions do
6          Objects = DataBase.FindObjectsBySession(Session);
7          foreach Object in Objects do
8              Type = Object.ObjectType;
9              TypesCounter[Type] += 1;
10         end
11     end
12     foreach KV in TypesCounter do
13         if KV.Value > LIMIT_OF_OBJECTS_SAME_TYPE then
14             Flag = True;
15         end
16     end
17     return Flag;
18 end
```

Algorithm 7: Algorithm for Feature 6.

```
Input:
DataBase – NoSQL DataBase with user's features
User – User for check the insider's feature
Output:
Flag – Boolean flag of insider's feature
1  begin
2      Flag = False;
3      Sessions = DataBase.FindSessionsByUser(User);
4      foreach Session in Sessions do
5          Objects = DataBase.FindObjectsBySession(Session);
6          foreach Type in TypeOfObjects do
7              ObjectsByType = Objects.FindByType(Type);
8              if FuzzyHash.CompareObjects(ObjectsByType) == True then
9                  Flag = True;
10             end
11         end
12     end
13     return Flag;
14 end
```

5.7. Algorithm for F_7

The feature of connections to external hosts can be determined by the SessionSites features of the Session, which contains the list of URLs. Also, to get a more complete picture, you can use the DeviceGeo feature of the Device from which the user accessed this URL (Algorithm 8).

Algorithm 8: Algorithm for Feature 7.

```
Input:
DataBase – NoSQL DataBase with user's features
User – User for check the insider's feature
Output:
Flag – Boolean flag of insider's feature
1  begin
2      Flag = False;
3      Sessions = DataBase.FindSessionsByUser(User);
4      foreach Session in Sessions do
5          Sites = Session.SessionSites;
6          foreach Site in Sites do
7              if SuspiciousSitesBase.IsExist(Site) == True then
8                  Flag = True;
9              end
10         end
11     end
12     return Flag;
13 end
```

The algorithm accepts a hybrid NoSQL database and a verified user as input. The output of the algorithm is the fact of confirming the feature of the insider. The principle of the algorithm is based on the following steps. First, all sessions of a given user are searched. Secondly, all sites visited by the user in this session are searched. In addition, thirdly, each of the sites is checked against the base for belonging to suspicious. In the case of a positive check, the algorithm returns confirmation of the stego-insider feature, otherwise – a refutation.

5.8. Algorithm for F_8

The feature of the presence or absence of encryption of the data transmitted by the user can be determined by two features: ObjectCrypto for Object and SessionCrypto for Session (Algorithm 9).

The algorithm accepts a hybrid NoSQL database and a verified user as input. The output of the algorithm is the fact of confirming the feature of the insider. The principle of the algorithm is based on the following steps. First, all sessions of a given user are searched. Secondly, all objects transmitted within the session are searched. In addition, thirdly, each session and its objects are checked for encryption. If found, the algorithm returns confirmation of the stego-insider feature, otherwise – a refutation.

5.9. Algorithm for F_9

The feature of the proximity of a potential insider's activity to a legal user is a qualitatively different problem with greater complexity. In this case, it is possible to use almost all the features stored in the database (Algorithm 10).

The algorithm accepts a hybrid NoSQL database and a verified user as input. The output of the algorithm is the fact of confirming the feature of the insider. The principle of the algorithm is based on calculating the similarity of user behavior to legal. If the similarity is less than the limit value, the algorithm returns confirmation of the stego-insider feature, otherwise – a refutation.

5.10. Stego-Insider Definition Scheme

The general scheme for determining the stego-insider is shown in Figure 4.

Algorithm 9: Algorithm for Feature 8.

```
Input:
DataBase – NoSQL DataBase with user's features
User – User for check the insider's feature
Output:
Flag – Boolean flag of insider's feature
1  begin
2      Flag = False;
3      Sessions = DataBase.FindSessionsByUser(User);
4      foreach Session in Sessions do
5          if Session.SessionCrypto == True then
6              Flag = True;
7          end
8          Objects = DataBase.FindObjectsBySession(Session);
9          foreach Object in Objects do
10             if Object.ObjectCrypto == True then
11                 Flag = True;
12             end
13         end
14     end
15     return Flag;
16 end
```

Algorithm 10: Algorithm for Feature 9.

```
Input:
DataBase – NoSQL DataBase with user's features
User – User for check the insider's feature
Output:
Flag – Boolean flag of insider's feature
1  begin
2      Flag = False;
3      Measure = UserBehavior.EvaluateSimilarity(User, LegalUser);
4      if Measure < LIMIT_OF_SIMILARITY_WITH_LEGAL_USER then
5          Flag = True;
6      end
7      return Flag;
8  end
```

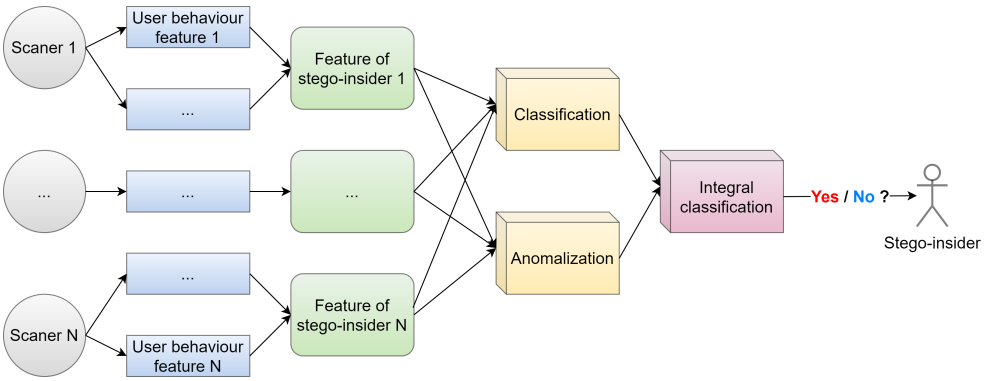

Figure 4. General scheme for determining the stego-insider.

According to the scheme, scanners first collect features of user actions. Then special algorithms reveal features in them. Each feature is checked both for anomalies in it and

for classifying it as "suspicious of creating hidden channels". The results obtained at the output are fed to the integral classifier, which already gives the final answer – whether this behavior is inherent in the stego-insider.

6. Experiment

To base the performance of the proposed model and algorithms, we present an experiment on a stand developed for this purpose, integrated into a real working network with many users.

The operating principle of the testbed was as follows. The organization's network traffic after the switch went to the ports of the virtual machines in the VMware vSphere (ESXi 6.5) environment. These virtual machines made up a coordinated cluster running CentOS 7 operating systems and running Apache Hadoop products with the Cloudera implementation–Apache Ambari. The cluster had the following characteristics: 18vCPU 1.2 GHz, 48 GB RAM, 1.2 TB HDD. Five cluster computers acted as a DataNode and were responsible for storing and processing data, one computer (NameNode) was a manager for distributing cluster tasks and loads, as well as for storing a table of file names. In addition, data was collected from the authentication, authorization and accounting server, which was deployed based on the Cisco ISE version 2.0 solution. The tested logic is shown in Figure 5.

The activity of stego-insiders was emulated by creating stego-channels to an external server using the software from Table 2 for various types of input and output data (text, images, sound, video). Then the information was transmitted via the created stego-channel. Thus, in addition to legal network traffic, the organization had packets that formed hidden channels. The parameters of the stego-channels were different and depended on the software used to emulate the activities of insiders. A total of 800 GB of network traffic was collected, generated by 200 users in the network into which the stand was installed. Of these, about 5 GB (i.e., about 0.5%) were network packets generated by an insider – containing the process of building a stego-channel and transmitting data over it.

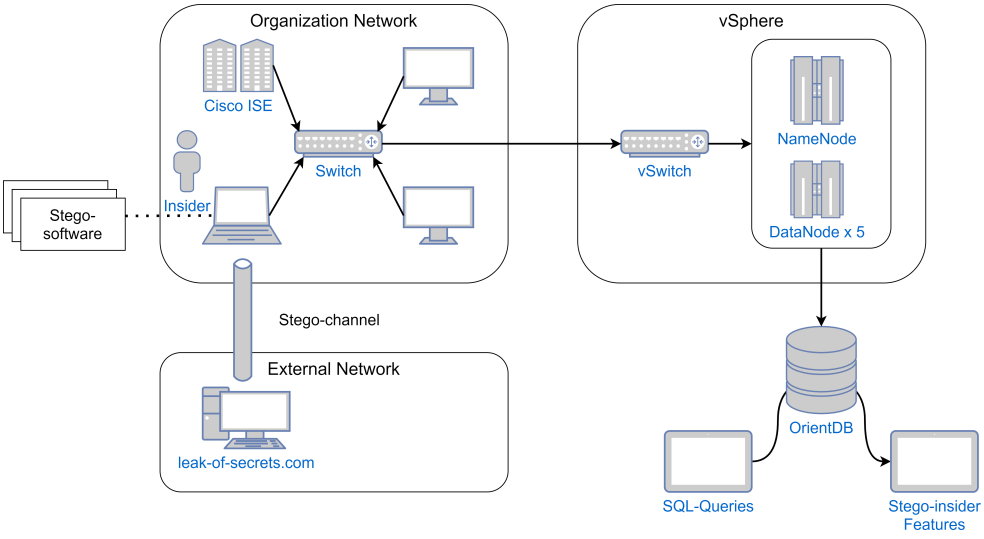

Figure 5. General scheme of experiment for determining the stego-insider.

In the experiment, all network traffic at the test bed was collected over a month – in the NoSQL Database. Then the data was manually checked, processed using SQL queries, and analyzed for the presence of features of stego-insiders by the above algorithms

To identify features of an insider, SQL queries were applied to the database, which did not require the development of additional software tools. As will be shown below, the results of the queries allowed us to partially identify the features of a stego-insider (user Alex in the examples below).

6.1. Environment

The implementation of the presented database model was carried out using the OrientDB DBMS. The database implementation allows SQL queries to be processed by returning information (in table-oriented view) that was used to predict the presence of feature of insider activity.

For the experiment, the network activity of a small network is entered into the database for several days.

Figure 6 shows the generated database, presented in graphical form. The figure shows sessions – blue circles (circles in the central part), as well as communication with users – red circles (five circles on the left), communication with applications – yellow circles (three circles above) and connections with devices – green circles (seven circles below). With the help of a graphical presentation, you can easily and conveniently display the necessary data for a specific user, session, application or device. Figure 7 shows a selection of the database for a unique session, as well as all users, applications and devices associated with this session.

In addition to the real behavior of users, the actions of stego-insiders were emulated in the network according to various scenarios. Then, using special queries from the database, information was obtained about user behavior (from the point of view of creating hidden channels), which made it possible to identify each of the features of an attacker. Database queries and analysis of their responses is given below. Let us denote requests with the R_N record (from the word Request), where N is a sequence number. Naturally, to classify a user as an intruder, it is necessary to obtain information about all the features (i.e., to fulfill all requests) that will make it possible to form a final opinion. All requests to the created and filled database are real. The answers are also real and slightly revised to be human-oriented.

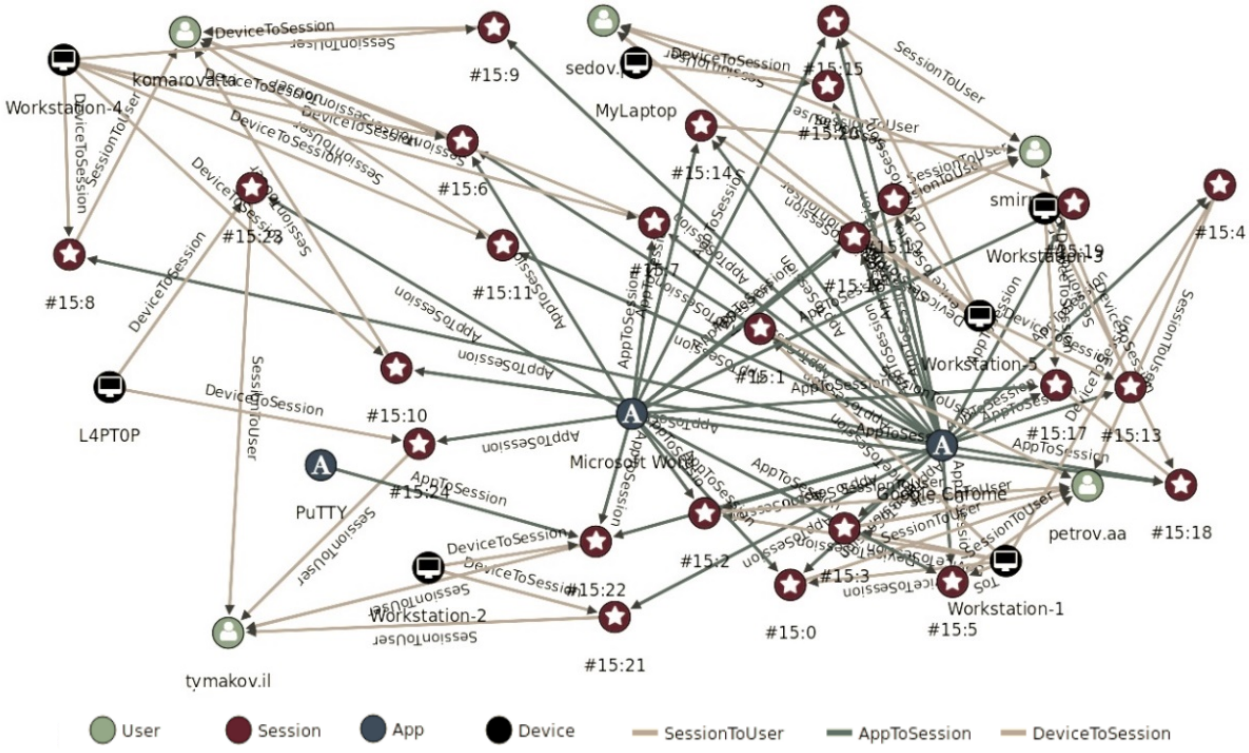

Figure 6. Graphical representation of the database.

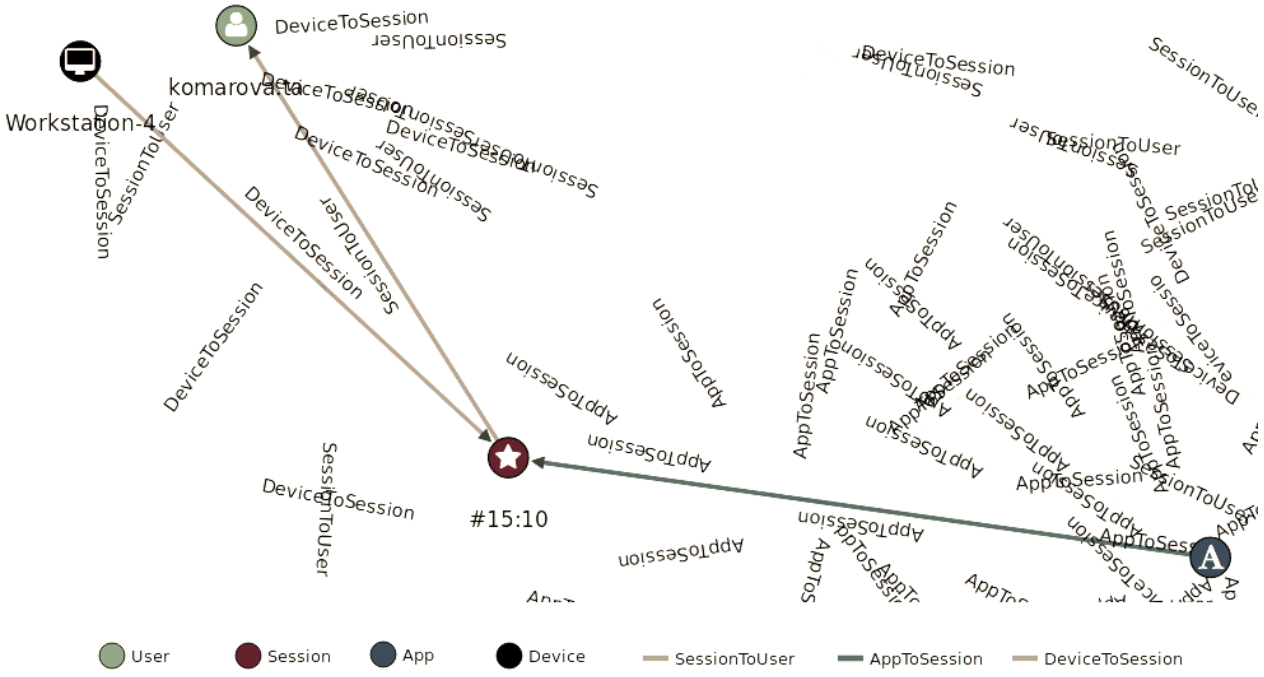

Figure 7. Sample for a specific session.

6.2. R_1. Steganographic Analysis Results

Using a query in the database, you can immediately get information about the potential presence of attachments to the transferred objects, obtained by using steganalysis. Let us give an example of such a request:

```
SELECT
OUT('SessionToUser').UserLogin AS Login,
OUT('DeviceToSession').DeviceName AS Device,
OUT Objects.ObjectStegoAnalysis AS Stego
FROM Session
```

The result of the query is shown in Table 6, which provides information about the username, device name, and the network application used.

Table 6. Example of query results of the algorithm for Feature 1.

Login	Device	Stego
Ivan	Workstation1	[No]
Fedor	Workstation2	[No, No, No]
Alex	Workstation3	[Yes, Yes, Yes, Yes, Yes, Yes]

Thus, the first user forwarded a file that was not marked as containing stego-embeddings. The second user similarly sent 3 files. However, for a third user, steganalysis gave the potential content of attachments in each of his six files. The last fact is a feature of stego-insider.

6.3. R_2. Availability of Software for Stego-Embeddings

Here is an example of a query that can be used to get a list of users, their devices and the network software used on them:

```
SELECT
```

```
OUT('SessionToUser').UserLogin AS Login,
OUT('DeviceToSession').DeviceName AS Device,
OUT('AppToSession').ApplicationName AS App
FROM Session
```

The result of the query is displayed in Table 7, which provides information about the username, device name, and the network application used.

Table 7. Example of query results of the algorithm for Feature 2.

Login	Device	App
Ivan	Workstation1	Chrome
Fedor	Workstation2	Internet Explorer
Alex	Workstation3	QuickStego

Thus, the first two users have "secure" applications on their devices, and the third has an application for embedding text messages into multimedia objects (see Table 1). The latter is a feature of a stego-insider.

6.4. R_3. Content of Embedding in IP

Here is an example of a request that can be used to obtain data from IPv4 fields that can potentially be used to form a stego-channel:

```
SELECT
OUT('SessionToUser').UserLogin AS Login,
OUT('DeviceToSession').DeviceName AS Device,
Packets.PacketIPv4Identifier AS IPv4Ident,
Packets.PacketIPv4DSCP AS IPv4Dscp,
Packets.PacketIPv4DNFBit AS IPv4NFBit,
FROM Session
```

The result of the query is displayed in Table 8, which provides information about the username, device name, and the network application used.

Table 8. Example of query results of the algorithm for Feature 3.

Login	Device	IPv4Ident	IPv4Dscp	IPv4NFBit
Ivan	Workstation1	...	...	...
Fedor	Workstation2	...	...	...
Alex	Workstation3	...	...	...

Note. For simplicity, the table does not indicate the exact content of the IPv4 packet fields (they are in binary format).

In this way, you can get a sequence of bits, "transmitted" using the fields of IPv4 packets, which can potentially be used to create a hidden channel. Identifying suspicious content in them (for example, using steganalysis) could be a feature of a stego-insider.

6.5. R_4. Content of Embedding in TCP

Here is an example of a request that can be used to obtain data from TCP fields that can potentially be used to form a stego-channel:

```
SELECT
OUT('SessionToUser').UserLogin AS Login,
OUT('DeviceToSession').DeviceName AS Device,
Packets.PacketTcpSrcPort AS TcpSrcPort,
FROM Session
```

The result of the query is displayed in Table 9, which provides information about the username, device name, and the network application used.

Table 9. Example of query results of the algorithm for Feature 4.

Login	Device	TcpSrcPort
Ivan	Workstation1	...
Fedor	Workstation2	...
Alex	Workstation3	...

Note. For simplicity, the table does not indicate the exact content of the TCP packet fields (they have a binary format).

Thus, you can get a sequence of bits "transmitted" using the fields of TCP packets, which can potentially be used to create a hidden channel. Identifying suspicious content in them (for example, using steganalysis) could be a feature of a stego-insider.

6.6. R_5. Number of Objects of Different Types

Here is an example of a query that can be used to get the number of objects of each type:

```
SELECT
OUT(‘‘SessionToUser’’).UserLogin AS Login,
OUT(‘‘DeviceToSession’’).DeviceName AS Device,
Objects.ObjectType AS Type,
FROM Session
```

The result of the query is shown in Table 10, which provides information about the username, device name, and site visits.

Table 10. Example of query results of the algorithm for Feature 5.

Login	Device	Type
Ivan	Workstation1	[unknown]
Fedor	Workstation2	[txt, video, audio]
Alex	Workstation3	[image, image, image, image, image, image, image]

As you can see from the results, the first user uploaded one file of unknown type. In theory, the file could contain embedded information, but a single transfer is not enough to create a hidden channel. The second user equally uses text, video and audio file transfers – which is not suspicious. However, the third user exchanges multiple images – which can be a feature of stego-insider. This situation will be strengthened if the degree of difference between the files (feature F_6) is minimal.

6.7. R_6. Difference of Objects

Here is an example of a query that can be used to find the proximity of the contents of the transmitted objects. To do this, you can get fuzzy hashes of all objects and compare them with each other. The first is done with the next request:

```
SELECT
OUT(‘SessionToUser’).UserLogin AS Login,
OUT(‘DeviceToSession’).DeviceName AS Device,
Objects.ObjectHash AS Hash,
FROM Session
```

The result of the query is displayed in Table 11, which provides information about the user login, device name and site visits.

Table 11. Example of query results of the algorithm for Feature 6.

Login	Device	Hash
Ivan	Workstation1	[..., ... ,...]
Fedor	Workstation2	[..., ... ,...]
Alex	Workstation3	[..., ... ,...]

Note. For simplicity, the table does not indicate the exact content of the Hash (it has a binary format).

The proximity of Fuzzy Hash will indicate the similarity of the contents of the objects (i.e., images, text, audio and video files) – this is a feature of a stego-insider.

6.8. R_7. List of Connections to External Hosts

Here is an example of a query that can be used to get a list of sites visited by the user:

```
SELECT
OUT('SessionToUser').UserLogin AS Login,
OUT('DeviceToSession').DeviceName AS Device,
SessionSites AS Sites,
FROM Session
```

The result of the query is shown in Table 12, which provides information about the username, device name, and site accesses.

Table 12. Example of query results of the algorithm for Feature 7.

Login	Device	Sites
Ivan	Workstation1	[resource1.internal.com, accessed on 25 March 2021]
Fedor	Workstation2	[google.com, yahoo.com, microsoft.com, accessed on 25 March 2021]
Alex	Workstation3	[leak-of-secrets.com, accessed on 25 March 2021]

Thus, the first user accessed internal resources, the second – to external legitimate, and the third-to external suspicious (let us assume that the Leak of Secrets site is used to transfer confidential data to third parties).

6.9. R_8. Encryption Type or Lack Thereof

Here is an example of a query that can be used to determine the use of encryption in network transmission. Using an SQL query, we display information about sessions, transmitted objects and encryption facts. The request will be as follows:

```
SELECT
OUT('SessionToUser').UserLogin AS Login,
OUT('DeviceToSession').DeviceName AS Device,
SessionCrypto AS SessionCrypto
Objects.ObjectCrypto AS ObjectCrypto,
FROM Session
```

The result of the query is displayed in Table 13, which provides information on the login method, user login, number of login attempts, device name and application used to login.

Table 13. Example of query results of the algorithm for Feature 8.

Login	Device	SessionCrypto	ObjectCrypto
Ivan	Workstation1	No	No
Fedor	Workstation2	Yes	No
Alex	Workstation3	No	Yes

Thus, the first user does not use encryption, the second only when creating sessions, the third only when transferring objects.

The results can signal the feature of a stego-insider or his level of preparedness.

6.10. R_9. Closeness to Legal Behavior

Here is an example of a request that can be used to determine the legal behavior of users. Using an SQL query, we will display information about sessions and associated users, devices and applications. The request will be as follows:

```
SELECT
SessionLoginType AS Type,
SessionCrypto AS Crypto
OUT('SessionToUser').UserLogin AS Login,
OUT('SessionToUser').UserTotalAuth AS Auth,
OUT('DeviceToSession').DeviceName AS Device,
OUT('AppToSession').ApplicationName AS App
FROM Session
```

The result of the query is displayed in Table 14, which provides information on the login method, user login, number of login attempts, device name, and application used to login.

Table 14. Example of query results of the algorithm for Feature 9.

Type	Crypto	Login	Auth	Device	App
Standart	No	Ivan	1	Workstation1	Chrome
Standart	No	Fedor	3	Workstation2	Internet Explorer
Standart	No	John	2	Workstation3	Chrome

Thus, all three users used the standard method of logging into the system, spending no more than 3 attempts. User sessions did not use encryption mechanisms.

Similarly, you can try to find potential insiders who tried to guess the password, while being physically outside the perimeter of the organization. To identify such an anomaly, you can use the following query, which looks for users with 3 times the average number of login attempts:

```
SELECT
SessionLoginType AS Type,
SessionCrypto AS Crypto
OUT('SessionToUser').UserLogin AS Login,
OUT('SessionToUser').UserTotalAuth AS Auth,
OUT('DeviceToSession').DeviceName AS Device,
OUT('AppToSession').ApplicationName AS App
FROM Session
WHERE Auth >
3*(SELECT AVG(OUT(''SessionToUser'').UserTotalAuth) from Session)
```

The query result is shown in Table 15.

Table 15. Example of query results of the algorithm for Feature 10.

Type	Crypto	Login	Auth	Device	App
Remote	Yes	Alex	10	Notebook1	Chrome

Thus, the user Alex logged in remotely (Type = Remote), making many attempts to enter his username and password (Auth = 10). He is a potential insider. Please note that in

these examples, we are talking about suspicious user behavior, without taking into account their attempts to create hidden channels (i.e., stego-insiders are not directly determined).

The results can signal the feature of a stego-insider or his level of preparedness.

7. Assessment of the Insider Detection Complex

The algorithm for F_9 feature is most difficult for implementation. In the interests of this, a separate research work and an experiment to detect insiders (not stego) by their behavior in the network were performed. The experiment was performed at a separate stand in the really operating network of the organization. More than 200 GB of network traffic was collected. In addition to the actions of legal employees, the traffic contained the actions of insiders using simple and complex attack scenarios.

A software package was implemented to compare user behavior with behavior of the insider. The operating principle of the complex uses two methods – based on expert rules and machine learning. The first method is based on the application of strict rules created by experts based on their experience. The essence of the second method is to use one of the basic classifiers – decision trees (DT), naive Bayesian classifier (NB), k-nearest neighbors method (k-NN), support vector machine (SVM); or their compositions – Majority Vote (PV), Weighted Vote (WV), Soft Vote (SV), and Adaboost. Features from a NoSQL database were taken as data for classification. The second method was trained on data from a real network, as well as a generator of two types of insider activity scenarios: simple, consisting of single actions, and complex, which is their sequence.

Thus, each of the methods, using their own models and algorithms, for each user on the network made assumptions about classifying him as an "insider". To make a single judgment, a combination of the results of the methods was chosen, obtained in one of the following ways:

1. *consolidation* – the result of the complex operation includes insiders detected by any of the methods;
2. *intersection* – the result of the complex operation includes insiders detected by both methods simultaneously;
3. *only the first* – the result of the complex operation includes insiders discovered only by the first of the methods;
4. *only the second* – the result of the complex operation includes insiders detected only by the second of the methods.

The number of results obtained by combinations of methods and classifiers of the second method is 25.

The essence of the experiment was to identify malicious actions of an insider among the network traffic of legal users. The quality of the insider detection method was assessed using a classical F-measure for all combination ways. Based on the test results for each type of scenario, the combination with the highest F-measure was selected. This combination can be used as the main one for the operation of the complex.

The test results for the combinations with the highest F-measure are shown in Table 16; the following notation is used for measures of the quality of the insider detection complex: r – completeness, p – precise, a – accuracy, e – error, f – F-measure.

As can be seen from the table, the highest F-measure value is achieved for both attack scenarios when the results of the methods intersect – 0.9963 and 0.95, respectively. However, for a simple scenario, the NB classifier showed the best result, and for a complex one – SV. Also, the indicators of the F-measure in the case of a complex attack scenario are quite different from each other: the difference between the minimum and maximum values of the F-measure is ~12%, and between the maximum F-measure and the closest one is ~7%.

Following from the obtained measures of the quality of the work of this complex, it can be used as an implementation of a part of the algorithm for determining the F_9 feature.

Table 16. Results of testing the complex for detecting insiders (partial).

Combination of Methods	Classifier of the Second Method	r	p	a	e	f
Simple attack scenario						
Method 1	—	0.9915	0.9902	0.9909	0.0092	0.9909
Method 2	PV	0.9967	0.9950	0.9959	0.0042	0.9958
Combining methods	Adaboost	0.9912	0.9911	0.9912	0.0089	0.9912
Intersection of methods	NB	0.9956	0.9970	0.9963	0.0037	0.9963
Complex attack scenario						
Method 1	—	0.90	0.80	0.84	0.16	0.85
Method 2	Adaboost	0.94	0.84	0.88	0.12	0.89
Combining methods	SV	0.99	0.81	0.88	0.12	0.89
Intersection of methods	SV	0.90	1.00	0.95	0.05	0.95

8. Discussion

The problem of the presence of Type I errors (i.e., omission of insiders) and Type II errors (i.e., incorrect identification of insiders) is partially solved by collecting seemingly redundant features (F_8 and F_9), which can only signal indirectly.

The features are poorly formalized in the sense that they cannot serve as a guaranteed definition of the stego-insider. This problem is solved by applying machine learning methods [51], as shown in the stego-insider identification diagram (see Figure 4). According to the scheme, to accurately determine the intruder of this type, it is advisable to use the classification and identification of anomalies based on the collected features.

The analysis (especially in real time) of network traffic requires working with a large amount of data collection, storage, processing. This problem is partially solved in the following ways. First, using a specialized hybrid NoSQL database. Secondly, the absence in the database documents of the full content of network traffic and transmitted objects–since only the features necessary to identify stego-insiders are stored. Thirdly, by potential using resource management for parallel databases [52].

An attacker can use a whole range of software that allows him to embed messages into the sent container. However, the dedicated list of software solutions for stego-embedding (see Table 1) in the amount of 42 allows us to claim that we have knowledge about most of them. As a further solution to this particular problem, we can assume the creation and periodic updating of the signature (or other) database of such tools, placing them on the same line with virus software.

In the case of encryption of stego-embedding, obviously, it is unlikely that the original transmitted message will be obtained. However, the initial task was to determine the fact of hidden transmission channels. Receipt of the original message is considered only additional information, which allows, among other things, to make a portrait of the offender, assess the damage caused, etc.

In a sense, the proposed features can be considered quite simple, although they are not as trivial as the features of user behavior. However, in the case of complex attacker scenarios (for example, receiving stego-embeddings from one workstation and sending from another), they may not be effective enough. The solution can be found by introducing new features determined by the database using a whole complex of SQL queries. It is also possible to add intelligent features that are not determined by strict rules, but with the help of intelligent agents with the possibility of self-organization [53]—analyzing the behavior of users and devices, assigning them trust levels and even, in some cases, correcting access rights.

Some of the features in the case of WSNs are less reliable for detecting stego-insiders. The reason for this lies in the “convenience” of this environment for building hidden channels by an intruder. To improve the efficiency of the proposed model and method in this environment, it is advisable to use more intelligent algorithms for determining the

features of a stego-insider, taking into account the complex relationships between user behavior features (i.e., those that are difficult to determine manually by an expert).

If an intruder uses non-standard data transmission methods, it is obvious that the behavioral features and features of a stego-insider will not always be correctly identified. However, the very fact of using such methods indicates some anomaly in the user's work, and, therefore, is suspicious (the situation is similar to encryption).

Following from the review of the relevant works, the aspect of detecting insiders creating stego-channels was considered for the first time. Therefore, a comparative analysis with other methods is not fulfilled here. However, further let us compare the proposed system with similar systems used to detect classical (not stego) insiders.

9. Comparison of Methods

Let us make a comparative analysis of the stego-insider detection system (using the proposed model and features) with existing modern analogs in terms of functionality. For comparison, we take the following criteria: A – applicability for determining hidden channels, B–using Big Data processing methods, C – using specialized databases, D – using machine learning methods. The results are shown in Table 17 (the following designations and points are used: "+" – the presence of a parameter in the work, 1 point; "+/−" – partial compliance with the parameter, 0.5 point; "−" – its absence, 0 point).

Table 17. Comparative analysis of the proposed system with existing analogs.

Insider Detection Systems	A	B	C	D	Score
Neural Networks-Aided Insider Attack Detection for the Average Consensus Algorithm [54]	−	+	-	+	2
Enterprise Insider Detection as an Integer Programming Problem [55]	−	−	+/−	+	1.5
Insider Threat Identification Using the Simultaneous Neural Learning of Multi-Source Logs [56]	+	+/−	+	+/−	3
Scenario-Based Insider Threat Detection From Cyber Activities [57]	+/−	+/−	-	+/−	2.5
A Novel Mechanism for Fast Detection of Transformed Data Leakage [58]	+/−	+	+/−	+	3
Malicious Insider Attack Detection in IoTs Using Data Analytics [59]	+/−	+/−	+/−	−	2.5
Proposed system	+	+	+	+	4

The analysis of the comparison results allows us to assert the advantage of the proposed system (4 points) over analogs (1.5, 2, 2.5 and 3 points).

10. Conclusions

This paper focuses on identifying insiders in an organization. At the same time, a feature of insiders (previously practically not considered) is the creation of stego-channels by them for hidden data transmission.

The work is based on the following ideas. First, collecting many qualitatively different features about user behavior on the web. Secondly, the storage of features in a hybrid NoSQL database especially designed for this, combining graph and document-oriented ones. Thirdly, using the features collected by the scanners using SQL queries, the selection of several stego-insider features. In addition, fourthly, the application of machine learning methods (in terms of classification and anomalization) for the final determination of the stego-insider.

During the research phase, the features of user behavior in the network were identified. Then, an insider model was developed that stores these features. Based on the features and reviews of the works, an implementation of the model in the form of a hybrid NoSQL database was created. Moreover, the features were formed that characterize various properties of an insider. The key points of the features were their focus on detecting stego-channels created by the intruder. For each feature, an algorithm was developed to identify it from the database. Then, an experiment was conducted to build and work with an insider

model on a stand embedded in a real network. For each feature, examples of SQL queries to the database and their results were offered. The results were also analyzed to identify the stego-insider. The experiment showed the efficiency of the proposed approaches.

In the interests of implementing the algorithm for determining the sign of the closeness of the stego-insider's behavior to the legal one, a hardware-software complex for detecting insiders has been developed based on expert rules and machine learning. Testing of the complex showed high indicators of its quality measures.

The possibilities of building stego-channels in WSNs based on the specifics of this environment is noted. The reasons for some decrease in the reliability of features for detecting a stego-insider, leading to errors of I and II types, are shown. A direction for solving this problem is also proposed.

The main task of the study was to build and study a model for storing data, check the convenience of working with it, debug the work of SQL queries, and correct algorithms. At the same time, the method of detecting classic insiders by their behavior in the network was separately investigated and tested in practice.

Further development of the work should be the introduction of machine learning (in terms of the final classification of users into legal and offenders) and a full-fledged implementation of the system for determining stego-insiders. At the same time, it is required to expand the set of stego-insider features, taking into account more complex scenarios of its behavior. It is advisable to pay special attention to the features of stego-insiders in WSNs, as one of the environments used in critical infrastructures that are susceptible to attacks by intruders and suitable for building hidden data transmission channels. Also, from a scientific and practical point of view, the creation of separate metrics for assessing legal users from the standpoint of their possible transition to the category of "stego-insiders" will be in demand. Creation of a full-fledged product, including operating in Real-Time mode is also planned for future studies.

Author Contributions: I.K. was responsible for conceptualization and methodology; A.K. and I.U. analyzed the data; K.I. conceived and designed the experiment; all authors wrote the paper. All authors have read and agreed to the published version of the manuscript.

Funding: The reported study was funded by Russian Ministry of Science (information security), project number 5/2020.

Data Availability Statement: Not applicable.

Acknowledgments: The reported study was funded by Russian Ministry of Science (information security), project number 5/2020.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mescheryakov, S.; Shchemelinin, D.; Izrailov, K.; Pokussov, V. Digital Cloud Environment: Present Challenges and Future Forecast. *Future Internet* **2020**, *12*, 82. [CrossRef]
2. Schneier, B. *Secrets & Lies: Digital Security in a Networked World*, 1st ed.; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2000.
3. Simmons, G.J. The Prisoners' Problem and the Subliminal Channel. In *Advances in Cryptology: Proceedings of CRYPTO '83*; Springer: Boston, MA, USA, 1983; pp. 51–67.
4. Walker-Roberts, S.; Hammoudeh, M.; Dehghantanha, A. A Systematic Review of the Availability and Efficacy of Countermeasures to Internal Threats in Healthcare Critical Infrastructure. *IEEE Access* **2018**, *6*, 25167–25177. [CrossRef]
5. Walker-Roberts, S.; Hammoudeh, M.; Aldabbas, O.; Aydin, M.; Dehghantanha, A. Threats on the horizon: Understanding security threats in the era of cyber-physical systems. *J. Supercomput.* **2020**, *76*, 2643–2664. [CrossRef]
6. Kotenko, I.; Krasov, A.; Ushakov, I.; Izrailov, K. Detection of Stego-Insiders in Corporate Networks Based on a Hybrid NoSQL Database Model. In Proceedings of the 4th International Conference on Future Networks and Distributed Systems (ICFNDS), St. Petersburg, Russia, 28 May 2020.
7. Vanelli, B.; Pereira da Silva, M.; Manerichi, G.; Sandro Roschildt Pinto, A.; Antonio Ribeiro Dantas, M.; Ferrandin, M.; Boava, A. Internet of Things Data Storage Infrastructure in the Cloud Using NoSQL Databases. *IEEE Lat. Am. Trans.* **2017**, *15*, 737–743. [CrossRef]
8. Küçükkeçeci, C.; Yazici, A. Multilevel Object Tracking in Wireless Multimedia Sensor Networks for Surveillance Applications Using Graph-Based Big Data. *IEEE Access* **2019**, *7*, 67818–67832. [CrossRef]

9. Roy, P.; Mazumdar, C. Modeling of Insider Threat using Enterprise Automaton. In Proceedings of the 2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT), West Bengal, India, 12–13 January 2018; pp. 1–4. [CrossRef]
10. Santos, E.E.; Santos, E.; Korah, J.; Thompson, J.E.; Murugappan, V.; Subramanian, S.; Zhao, Z. Modeling insider threat types in cyber organizations. 2017 IEEE International Symposium on Technologies for Homeland Security (HST), Greater Boston, MA, USA, 25–26 April 2017; pp. 1–7. [CrossRef]
11. Igbe, O.; Saadawi, T. Insider Threat Detection using an Artificial Immune system Algorithm. In Proceedings of the 2018 9th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON), New York, NY, USA, 8–10 November 2018; pp. 297–302. [CrossRef]
12. Sav, U.; Magar, G. *Insider Threat Detection Based on Anomalous Behavior of User for Cybersecurity. Data Science and Security*; Jat, D.S., Shukla, S., Unal, A., Mishra, D.K., Eds.; Springer: Singapore, 2021; pp. 17–28.
13. Almazaydeh, W.I.A.; Sheshadri, H.S. Image Steganography Using a Dynamic Symmetric Key. In Proceedings of the 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 11–12 May 2018; pp. 1507–1513. [CrossRef]
14. Avci, D.; Tuncer, T.; Avci, E. A new information hiding method for audio signals. In Proceedings of the 2018 6th International Symposium on Digital Forensic and Security (ISDFS), Antalya, Turkey, 22–25 March 2018; pp. 1–4. [CrossRef]
15. Bedi, P.; Dua, A. Network Steganography Using Extension Headers in IPv6. In *Information, Communication and Computing Technology*; Badica, C., Liatsis, P., Kharb, L., Chahal, D., Eds.; Springer: Singapore, 2020; pp. 98–110.
16. You, W.; Zhang, H.; Zhao, X. A Siamese CNN for Image Steganalysis. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 291–306. [CrossRef]
17. Smolarczyk, M.; Szczypiorski, K.; Pawluk, J. Multilayer Detection of Network Steganography. *Electronics* **2020**, *9*, 2128. [CrossRef]
18. Krauß, C.; Schneider, M.; Eckert, C. On handling insider attacks in wireless sensor networks. *Inf. Secur. Tech. Rep.* **2008**, *13*, 165–172. [CrossRef]
19. Krauß, C. Handling Insider Attacks in Wireless Sensor Networks. Ph.D. Thesis, Technische Universit'at, Darmstadt, Darmstadt, Germany, 2010.
20. Choi, J.; Bang, J.; Kim, L.; Ahn, M.; Kwon, T. Location-Based Key Management Strong Against Insider Threats in Wireless Sensor Networks. *IEEE Syst. J.* **2017**, *11*, 494–502. [CrossRef]
21. Meng, W.; Li, W.; Su, C.; Zhou, J.; Lu, R. Enhancing Trust Management for Wireless Intrusion Detection via Traffic Sampling in the Era of Big Data. *IEEE Access* **2018**, *6*, 7234–7243. [CrossRef]
22. Janarthanan, A.; Kumar, D.; Antony, R.R.; Parvathe, C.B.D. IADF security: Insider attack detection using fuzzy logic in wireless multimedia sensor networks. *Soft Comput.* **2020**, *24*, 13893–13902. [CrossRef]
23. Gu, Y.; He, C.; Liu, F.; Ye, J. Raman Ink for Steganography. *Adv. Opt. Mater.* **2021**, *9*, 2002038. [CrossRef]
24. Wang, S.; Zheng, N.; Xu, M. A Compression Resistant Steganography Based on Differential Manchester Code. *Symmetry* **2021**, *13*, 165. [CrossRef]
25. Liu, F.; Zhou, X.; Yan, X.; Lu, Y.; Wang, S. Image Steganalysis via Diverse Filters and Squeeze-and-Excitation Convolutional Neural Network. *Mathematics* **2021**, *9*, 189. [CrossRef]
26. Järpe, E.; Weckstén, M. Velody 2—Resilient High-Capacity MIDI Steganography for Organ and Harpsichord Music. *Appl. Sci.* **2021**, *11*, 39. [CrossRef]
27. Xiang, L.; Yang, S.; Liu, Y.; Li, Q.; Zhu, C. Novel Linguistic Steganography Based on Character-Level Text Generation. *Mathematics* **2020**, *8*, 1558. [CrossRef]
28. Korzhik, V.; Alekseev, V.; Morales-Luna, G. Audio watermarking system resistant to removal attacks by dereverberation. *Int. J. Comput. Sci. Appl.* **2018**, *15*, 1–15.
29. Kwak, M.; Cho, Y. A Novel Video Steganography-Based Botnet Communication Model in Telegram SNS Messenger. *Symmetry* **2021**, *13*, 84. [CrossRef]
30. Taleby Ahvanooey, M.; Li, Q.; Hou, J.; Rajput, A.R.; Chen, Y. Modern Text Hiding, Text Steganalysis, and Applications: A Comparative Analysis. *Entropy* **2019**, *21*, 355. [CrossRef] [PubMed]
31. Krasov, A.; Arshinov, A.; Ushakov, I. Embedding the hidden information into java byte code based on operands' interchanging. *ARPN J. Eng. Appl. Sci.* **2018**, *13*, 2746–2752.
32. Sharikov, P.; Krasov, A.; Gelfand, A.; Kosov, N. *Research of the Possibility of Hidden Embedding of a Digital Watermark Using Practical Methods of Channel Steganography*; Springer: Cham, Switzerland, 2020; pp. 203–209. [CrossRef]
33. Mushenko, A.; Dzuba, J.; Nekrasov, A.; Fidge, C. A Data Secured Communication System Design Procedure with a Chaotic Carrier and Synergetic Observer. *Electronics* **2020**, *9*, 497. [CrossRef]
34. Liu, J.; Tian, H.; Chang, C.C.; Wang, T.; Chen, Y.; Cai, Y. Steganalysis of Inactive Voice-Over-IP Frames Based on Poker Test. *Symmetry* **2018**, *10*, 336. [CrossRef]
35. Sun, C.; Tian, H.; Chang, C.C.; Chen, Y.; Cai, Y.; Du, Y.; Chen, Y.H.; Chen, C.C. Steganalysis of Adaptive Multi-Rate Speech Based on Extreme Gradient Boosting. *Electronics* **2020**, *9*, 522. [CrossRef]
36. Korzhik, V.; Nguyen, C.; Fedyanin, I.; Luna, G. Side Attacks on Stegosystems Executing Message Encryption Previous to Embedding. *J. Inf. Hiding Multimed. Signal Process.* **2020**, *11*, 44–57.

37. Atzeni, P.; Bugiotti, F.; Cabibbo, L.; Torlone, R. Data modeling in the NoSQL world. *Comput. Stand. Interfaces* **2020**, *67*, 103149. [CrossRef]
38. Venkatraman, S.; Fahd, K.; Kaspi, S.R.V. Versus NoSQL Movement with Big Data Analytics. *Int. J. Inf. Technol. Comput. Sci.* **2016**, 59–66. [CrossRef]
39. Davoudian, A.; Chen, L.; Liu, M. A Survey on NoSQL Stores. *ACM Comput. Surv.* **2018**, *51*, 1–43. [CrossRef]
40. Phiri, H.; Kunda, D. A Comparative Study of NoSQL and Relational Database. *Zambia ICT J.* **2017**, *1*, 1–4. [CrossRef]
41. Abourezq, M.; Idrissi, A. Database-as-a-Service for Big Data: An Overview. *Int. J. Adv. Comput. Sci. Appl.* **2016**, *7*. [CrossRef]
42. Priyanka, A. A Review of NoSQL Databases, Types and Comparison with Relational Database. *Int. J. Eng. Sci. Comput.* **2016**, *6*, 4963–4966.
43. Kotenko, I.; Fedorchenko, A.; Doynikova, E.; Chechulin, A. An Ontology-based Storage of Security Information. *Inf. Technol. Control* **2018**, *47*, 655–667. [CrossRef]
44. Shterenberg, S.; Maria, P. A Distributed Intrusion Detection System with Protection from an Internal Intruder. *Autom. Control. Comput. Sci.* **2018**, *52*, 945–953. [CrossRef]
45. Kotenko, I.; Saenko, I.; Kushnerevich, A. Architecture of the Parallel Big Data Processing System for Security Monitoring of Internet of Things Networks. *SPIIRAS Proc.* **2018**, *4*, 5. [CrossRef]
46. Kotenko, I.; Kuleshov, A.I.; Ushakov, I. A system for collecting, storing and processing security information and events based on elastic stack tools. *SPIIRAS Proc.* **2017**, *5*, 5–34. [CrossRef]
47. Zheng, Y.; Liu, F.; Yang, C.; Luo, X.; Zhao, K. Identification of Steganography Software Based on Core Instructions Template Matching. In Proceedings of the 2011 Third International Conference on Multimedia Information Networking and Security, Shanghai, China, 4–6 November 2011; pp. 494–498. [CrossRef]
48. Zheng, Y.; Liu, F.; Luo, X.; Yang, C. A Method Based on Feature Matching to Identify Steganography Software. In Proceedings of the 2012 Fourth International Conference on Multimedia Information Networking and Security, Nanjing, China, 2–4 November 2012; pp. 989–994. [CrossRef]
49. Namanya, A.P.; Mirza, Q.K.A.; Al-Mohannadi, H.; Awan, I.U.; Disso, J.F.P. Detection of Malicious Portable Executables Using Evidence Combinational Theory with Fuzzy Hashing. In Proceedings of the 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), Vienna, Austria, 22–24 August 2016; pp. 91–98. [CrossRef]
50. Naik, N.; Jenkins, P.; Savage, N. A Ransomware Detection Method Using Fuzzy Hashing for Mitigating the Risk of Occlusion of Information Systems. In Proceedings of the 2019 International Symposium on Systems Engineering (ISSE), Edinburgh, UK, 1–3 October 2019, pp. 1–6. [CrossRef]
51. Balueva, A.; Desnitsky, V.; Ushakov, I. *Approach to Detection of Denial-of-Sleep Attacks in Wireless Sensor Networks on the Base of Machine Learning*; Springer: Cham, Switzerland, 2020; pp. 350–355. [CrossRef]
52. Tan, Z.; Babu, S. Tempo: Robust and Self-Tuning Resource Management in Multi-Tenant Parallel Databases. *Proc. VLDB Endow.* **2016**, *9*, 720–731. [CrossRef]
53. Walker-Roberts, S.; Hammoudeh, M. Artificial Intelligence Agents as Mediators of Trustless Security Systems and Distributed Computing Applications. In *Guide to Vulnerability Analysis for Computer Networks and Systems: An Artificial Intelligence Approach*; Parkinson, S., Crampton, A., Hill, R., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 131–155. [CrossRef]
54. Li, G.; Wu, S.X.; Zhang, S.; Li, Q. Neural Networks-Aided Insider Attack Detection for the Average Consensus Algorithm. *IEEE Access* **2020**, *8*, 51871–51883. [CrossRef]
55. Kavun, S.; Sorbat, I.; Kalashnikov, V. Enterprise Insider Detection as an Integer Programming Problem. In *Intelligent Decision Technologies. Smart Innovation, Systems and Technologies*; Watada J., Watanabe, T., Phillips-Wren, G., Howlett, R., Jain, L., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 16._29. [CrossRef]
56. Liu, L.; Chen, C.; Zhang, J.; De Vel, O.; Xiang, Y. Insider Threat Identification Using the Simultaneous Neural Learning of Multi-Source Logs. *IEEE Access* **2019**, *7*, 183162–183176. [CrossRef]
57. Chattopadhyay, P.; Wang, L.; Tan, Y. Scenario-Based Insider Threat Detection From Cyber Activities. *IEEE Trans. Comput. Soc. Syst.* **2018**, *5*, 660–675. [CrossRef]
58. Huang, X.; Lu, Y.; Li, D.; Ma, M. A Novel Mechanism for Fast Detection of Transformed Data Leakage. *IEEE Access* **2018**, *6*, 35926–35936. [CrossRef]
59. Khan, A.Y.; Latif, R.; Latif, S.; Tahir, S.; Batool, G.; Saba, T. Malicious Insider Attack Detection in IoTs Using Data Analytics. *IEEE Access* **2020**, *8*, 11743–11753. [CrossRef]