IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 9 August 2022, accepted 14 August 2022, date of publication 18 August 2022, date of current version 26 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3199749

APPLIED RESEARCH

StegFog: Distributed Steganography Applied to Cyber Resiliency in Multi Node Environments

JĘDRZEJ BIENIASZ, PATRYK BĄK, AND KRZYSZTOF SZCZYPIORSKI, (Senior Member, IEEE)
Institute of Telecommunications, Warsaw University of Technology, 00-665 Warszawa, Poland

Corresponding author: Jędrzej Bieniasz (Jedrzej.Bieniasz@pw.edu.pl)

This work was supported by the Polish National Centre for Research and Development under Project CYBERSECIDENT/369234/I/NCBR/2017.

ABSTRACT This paper presents a new study about the communication system named StegFog. It is based on applying distributed steganography linked with the concept of cyberfog security. The idea of cyberfog is to secure data by splitting it into numerous fragments and spreading them to end-user devices. The design of StegFog combines steganographic methods previously investigated by the Authors – StegHash and SocialSteg Disc. The reference design of the system is proposed and analyzed in terms of security, reliability, performance and timing. To prove the feasibility of the proposed solution, the prototype of StegFog was implemented, consisting of steganographic nodes operating over BitTorrent P2P networks with a hidden communication channel established by the multimedia files shared within it. Experiments were conducted for the implemented prototype to understand the basic features of the system, such as operational aspects of the address generator constructed using the modified StegHash method, timing of the basic methods within the system – reading, writing and notifying, steganographic rate of the system and overall security provided by the system. The presented results show how system such as StegFog aligned with the cyberfog security concept could be considered as the protection measure for secure network communications. The practicality, challenges and research directions to extend the concept are also discussed.

INDEX TERMS Cyberfog, distributed steganography, fog computing, information hiding, network security, StegHash, SocialStegDisc, peer-to-peer networks.

I. INTRODUCTION

Recently, the main computing paradigms have been rapidly evolving. Centralized computing is generally developing as Cloud Computing and Data Centers with involving some distribution and redundancy. Distributed computing is mostly realized with Edge Computing and Internet of Things (IoT) applications. The examples of such smart and interconnected devices are mobile phones, wearables, sensors, power grids, etc. As both of these computing solutions are solving many industrial and life use-cases, there is a highly need for the methods complementing them, especially in case of cyber-security. One of such emerging ideas is fog computing, which is recognized by [1], [2] as the area of developing a new solutions for network security, data security or privacy. Cyberfog as one of the concepts of combining fog computing and cybersecurity was recently introduced by [3]. Such approach is characterized with higher dispersion and distribution of data that establishes cyber attack resiliency. In the case of a data breach, an adversary could steal only part of the data from the system. Furthermore, such data would be structurally dispersed and without knowing the secret, the attack would not have achieved its goals. Such idea assumes that security of networks and data would increase by *maximizing the "fogginess" of information* [3]. Whereas these benefits are understandable, there are numerous challenges of cyberfog such as management complexity, network bandwidth, memory consumption, power demand, latency and connectivity.

The presented characteristics of cyberfog is corresponding with Authors' previous research on distributed steganography. One of the milestones was the concept of TrustMAS introduced by [4]. TrustMAS is described at Related Work (Section II). Recently, the idea of text steganography [5] has

The associate editor coordinating the review of this manuscript and approving it for publication was Byung-Seo Kim.

been combined with social networks into new distributed information hiding methods. StegHash [6] is the method to use multimedia objects as carriers of hidden information and to disperse them across open social networks (OSNs). A logical connection between them is established by means of a mechanism of hashtags as the text markers in the form of #<tag>, are commonly applied in OSNs. In addition, the Author of StegHash proposes an option for applying the StegHash technique to create a steganographic filesystem analogous to the existing classic filesystems, such as FAT (File Allocation Table) or NTFS (New Technology File System). SocialStegDisc [7] follows up the StegHash research by proposing the application of basic concepts of classic filesystems, such as create-read-update-delete operations or defragmentation processes to the original idea of StegHash. Furthermore, time-memory trade-offs were proposed by the design. The summarized research results were presented in [8].

The motivation of this work is to propose a new distributed steganographic system for protecting networks conformed with cyberfog and tackling its challenges. The system needs to be practically implementable with the end-user devices and to offer acceptable tradeoff between availability and confidentiality of data. A combination of cyberfog with distributed steganography was firstly investigated by the Authors in [9]. This paper introduces a new method called StegFog following the theoretical ideas from [9].

StegFog utilizes the indexing scheme introduced by StegHash [6] and provides basic operations like in SocialStegDisc [7]. The design of the system expands them by using the end-users' devices' memory as a carrier for objects with hidden data. The methods of data fragments routing and finding the next parts among such peer-to-peer networks are also proposed. The working prototype of a such system is implemented for testing purposes. Then, it is evaluated in accordance to the challenges defined by [3], especially data fragments distribution, network management complexity, data transfer speed, endpoint disk capacity and computing power consumption. The security of StegFog is mainly ensured by the character of logical connection between the distributed parts of the data. The system could only be properly compromised when the adversary takes over the secret generation function. Without it, the captured parts, e.g., by compromising one or part of a device, would be unusable, thereby fulfilling the basic cyberfog requirement.

The main contributions of this paper are:

1) Further development of the initial concept presented in the paper [9] into a new distributed steganography method called StegFog with defining the functional and operational architecture, the underlying communication protocol with all required supporting functions and providing the security evaluation of the design.
2) Implementation of StegFog prototype to experiment on the aspects such as timing of operations, fragments management and security of the addressing scheme, reliability and network bandwidth utilization.

The structure of the paper is as follows:

- Section II briefly presents the development of the distributed steganography concept towards practical realizations and implications in the area of using it with cyber attackers in the wild. Then, it discusses the fog computing and cyberfog security concept and how it could be connected with distributed steganography.
- Section III defines the StegFog reference concept and architecture as the extension of the initial idea presented in [9]. The expanded security analysis is also conducted.
- Section IV shows how the prototype of StegFog was developed and implemented based on the reference design introduced in Section III.
- Section V evaluates the prototype of StegFog implemented for the purpose of this research through a few experiments to understand the basic features of StegFog in practical application.
- Section VI concludes the paper with a summary of the results and further research directions that could be developed from this paper.

II. RELATED WORK

A. DISTRIBUTED STEGANOGRAPHY

The term steganographic communication usually covers the exchange of messages via a steganographic channel between one sender and one receiver. For the first time, the term distributed steganography was used in [10], in 2011. Distributed steganography is characterized by communication through a hidden channel between several independent senders and one receiver. In [10] three examples of the use of distributed steganography are given:

- Military use – used by spy networks to communicate over untrusted networks in foreign countries.
- Business use – some investing projects could involve competing companies. Companies should not know about the financial contributions from other companies and the fact they are participating in a fundraiser.
- Knowledge Management use – when individual units should not be aware of the existence of secret information from other units, but it is needed to pull such information together.

In each of the referred examples, the advantage of steganography over cryptographic solutions is hiding the communication between the recipient and the senders. In these cases, any potential observer has no knowledge of the number of parties sharing information, nor are they able to establish their identity.

An example opposite to that mentioned in [10] is when one sender shares information over a steganographic channel with multiple recipients. This approach was named Broadcast Steganography and presented in [11]. In [11] the authors additionally used encryption to make the hidden message readable only to the authenticated users. To illustrate the use of steganographic broadcasting, the authors used the following examples:

- Military application. To detect internal threats, a secret investigative group would use a hidden channel for communication. Thus, the observer is not aware of an ongoing investigation, nor is he able to identify the investigating group.
- Human Rights activism. An activist can bypass censorship or, by skillful authentication of user groups, detect members infiltrating the association.

In [12] the Authors attempted to define a distributed steganographic system. According to the paper, a distributed steganographic system can be described by the following equation:

$$Pr_r[DSD([C]_s, [h]_s, DSE(r, [C]_t, [h]_t, m, e) < C_s >) = m] \geq 1 - \mu(k) \tag{1}$$

where:

- DSE, DSD - pair of probabilistic algorithms
- r - seed
- $[C]_t$ - target set of steganographic channels
- $[h]_t$ - historical log of steganographic channels
- m - message to be hidden
- e - minimal number of channels to preserve the message
- $[C]_s$ - the overt environment
- $[h]_s$ - set of historical channels

According to the Equation 1, a steganographic distributed system should, with a probability greater than or equal to $1 - \mu(k)$, allows the message to be restored using no less than e steganographic channels.

The idea of steganographic communication in a distributed system arose before the definition of a distributed steganographic system was proposed. In 2007, a distributed steganographic router operating in a multi-agent environment was introduced in [4]. The authors focused on ensuring the anonymity of network agents and ensuring hidden communication between them. Steganographic agents are designed to conduct covert communication in more than one layer of the TCP / IP model. In the TrustMAS platform, apart from ordinary agents using the anonymity function, steganographic agents are distinguished as StegAgents (SA). Behavior consistent with the definition of distributed steganography is the discovery process followed by SAs, where once a message is sent, it can reach a target node from more than one intermediate node.

Another area of development in the domain of distributed steganography is combination the concept with a popular Internet services. One of the target environments for such applications are Peer-to-Peer (P2P) network protocols. Such idea is summarized in [13], especially concentrating on application of BitTorrent protocol [14]. The described method of network steganography based on BitTorrent uses the modification of the order of the network packets in the data exchange protocol between nodes. It can be recognized as the timing method of network steganography applied on BitTorrent protocol message order.

The new distributed steganography methods are also developed around cloud computing. The paper [15] is proposing a distributed steganographic filesystem based on computation clusters. The main idea behind the method presented there consists in using multiple files to encode hidden information by means of their relative positions in clusters. Recently, the idea of using cloud computing resources was revisited by papers [16] for multi-cloud setup and its developed version [17] for single-cloud configuration. Distributed steganography was examined as an approach to hidden data in several files featuring leaving less traces than the classical methods. The method proposed in [16] offered the approach to steganography in multi-cloud storage environment with preservation of the integrity of files used as the hidden data carriers. However, such approach requires the management of several cloud storage environments and a more complex scheme to preserve secrecy of the steganographic protocol. [17] improves it by using a single cloud storage environment and reducing the complexity of secrecy management.

Another research results with using cloud storage capabilities for distributed steganography are presented in [18]. The main difference from the previous cited works is that it focuses on distributed steganography method at the level of hidden data carriers (e.g. image files) and storing it within a given system. It proposes an adaptive payload distribution for multiple images steganography stored in a cloud environment. For that purpose, Authors designed two payload distribution strategies based on image texture complexity and distortion distribution. They also provided a theoretical security analysis from the steganalyst's point of view.

B. FOG COMPUTING AND SECURITY TOWARDS CYBERFOG CONCEPT

The increasing adaption of the Internet of Things (IoT) solutions and applications outcomes as that a cyberspace is ever more connected, especially with the heterogeneous and mobile devices. The cyberspace of cyber-physical systems (CPS) [19] on one hand is producing a lot of data via sensors, actuators or RFID/NFC tags. On the other hand many end-user devices, such as smartphones, tablets, and PCs are consuming such data to make decisions and actions. Many components, standards and protocols must cooperate together to efficiently gather, process, and share such data. The fog computing [1], distributed by default, represents a promising solution which offers interoperability, scalability, security, and privacy. It operates as a middle layer between data consumers/producers and traditional cloud computing environments. Authors of [2] are analyzing the evolution in modeling the new methodologies related to fog computing and IoT. Paper shows how moving security and privacy tasks toward the edge of the network provide both advantages and new challenges to be faced in this research field.

One of the proposed solutions of how to apply fog computing for security is the cyberfog security approach was introduced by Kott *et al.* in [3]. They proposed to use fog computing as a method for mitigating a cyber adversary with uncertainty could provide greater attack resiliency. One of the functions to ensure security in the cyberfog approach is the division of valuable information into parts and their dispersion among different devices. Prof. Kott referred to the division of the secret according to Shamir's pattern [20]. This behavior is intended to increase the resilience of the system in case parts of the network are compromised. It assumes that the adversary should not be able to recreate the secret with fewer parts of the secret than the assumed threshold allowing for its complete reconstruction. The dispersed nature of the system also means that the attacker of such a system will have a difficult task, because to disable it, it will be necessary to take over or destroy a large number of devices that can be physically located in different places. To ensure data availability in a changing network, Prof. Kott additionally proposed the use of data fragment replication, known from database products, such as Apache Cassandra. Thanks to replication, it will be possible to recreate the required number of secret fragments despite the compromised part of the network.

The challenges with such an approach are mentioned in [3] and include a few aspects. The first one is how to distribute fragments of a secret. Fragments should be divided and distributed in the system in a way that prevents a potential adversary from knowing how they are dispersed to predict the addressees of the following parts. Another challenge is a situational awareness. Not all data fragments are equally important in the context, so that the data should be divided in such a way that the potential intercepted data fragments are of little significance to the adversary. The authors of the cyberfog concept also suggest obfuscating the information contained in the scattered fragments so that, if intercepted, there would be several equally probable interpretations of them. Going a step further, the data can be additionally secured so that, without knowing how to read them, they could lead the adversary to misinterpreting them. These mechanisms can be implemented not only at the information level, but also at the level of network traffic or network topology. More general challenges also include the level of network management complexity, data transfer speed, limiting endpoint disk capacity and computing power consumption.

C. StegFog IN CONTEXT OF THE RELATED WORK

As it was mentioned in Introduction (Section I), a combination of fog computing and cyberfog security with distributed steganography was firstly investigated by the Authors of this paper in [9]. They provided the idea of how to combine their previous works into a new way of secure communications aligned with the cyberfog concept. Based on the related work walk-through, there is a need of designing, implementing and validating a new security mechanisms in fog computing approach. This paper complements the current state-of-the-art in distributed steganography research by the comprehensive design of a new method characterized as the hybrid one and applied for the rising paradigm of fog computing. It further combines three types of classic steganography approaches into the one distributed steganography method StegFog:

- using any file steganography method, e.g image steganography on at *data-at-rest* level
- using text-based steganography method to hiddenly manage pointers to hidden data fragments (*data-at-rest* and *data-in-transit* level)
- using network protocol steganography on *data-in-transit* level

The paper is also presenting the implementation of a working prototype, which could be considered as one of the first working systems aligned with cyberfog approach defined by [3]. Then, it made possible to empirically measure several features of the system, such as as data fragments distribution security, network management complexity, data transfer speed or required endpoint disk capacity.

III. REFERENCE ARCHITECTURE OF StegFog SYSTEM

The idea behind this paper is to extend examination if the StegHash [6] and SocialStegDisc [7] mechanisms can be apply for communication systems for the fog in the devices. The paper [9] introduced the idea which is reminded by Section III-A. Section III-B generalizes the layered architecture of StegFog beyond the one presented in [9]. Section III-C evaluates security aspects and requirements of StegFog system. It includes the conclusions formulated in [9] and develop it further with [21] security assessment approach.

A. StegHash/SocialStegDisc APPLIED FOR StegFog

In StegFog there is no public and open internet services, like social networks in the original idea for both StegHash and SocialStegDisc. Instead, the system would operate between a defined set of end users' devices connected into Wide Area/Local Area networking setup. The next new element is defining the size of the problem. Previously, indexing the services that carry the uploaded steganograms could be indicated by one of the n hashtags in an index. In StegFog it is impossible, as the number of carrying devices increases many times. It could be approached by assuming that n devices would create a *memory sector* of size n. For each memory sector the mechanism of addressing the carrier of the data (end-user device) by one of the n hashtags would still be applicable within the memory sector. The tradeoff in such a concept is that the system would need a new layer for the communication on the level of the memory sectors. Every device would be locally associated with one memory sector from one of the many available in the system. It means that every device has its own instance of a memory controller to serve all the basic filesystem operations inside the memory sector working with algorithms introduced by StegHash/SocialStegDisc. During sending or storing data in the system, such a controller disperses fragments between n devices. Every device needs to have a copy of an allocation table for any memory sector and to support multiple membership devices in the memory sectors. As part of the design of the system for data storage and access schemes between the memory sectors are considered:

- Access through memory sector gateways. When a device saves the data to a memory sector it would notify the other devices that starting from the selected index there is data of B bytes stored. The storing device is selected as a gateway to access the stored data. This approach combines methods of routing and the file allocating methods. This approach introduces a synchronization of allocated addresses across the memory sectors and a state of generation function within the memory sector. Finally, the other devices have a global view of the allocated data.
- Direct access to read from any memory sector. In comparison to the first proposition, the routing information for uploaded data requires extensions to have all n device identifiers with an algorithm to select the following devices for retrieving data. The routing records are shared with every single upload to the system and cached by devices (proactive way).
- Hybrid approach of 1) and 2). It assumes sharing only metadata of the file and a selected gateway with every single upload to the system. When the data is retrieved, at first the request for the memory scheme is issued to the gateway. The gateway responds back with all n devices' identifiers and an identification scheme. Next, the requested part of the data could be read directly from the memory sector.

B. StegFog LAYERED ARCHITECTURE

Generally, the system needs to provide two network services for communication:

- Sending data end-to-end between network nodes. It could be provided by TCP/IP transport layer protocols – TCP or UDP. The other protocols from upper layers could be also considered.
- Reachability between network nodes. The devices within the system need to establish a network, so routing is necessary. Network addresses could be distributed with a static routing configuration or through a dynamic routing protocol.

The described network communication should be built upon a platform that offers hidden distributed communication while preserving privacy. Any device in StegFog would work as a *StegPeer* utilizing steganographic components to provide covert channels to realize communication. The concept of StegFog expands beyond the original ideas [6], [7] by:

- Using the end-users' devices' memory as a carrier for objects with hidden data.
- Using any type of object as a carrier of hidden data.
- Data could also be written directly into the end-users' device's memory by embedding it into a carrier.

The proposed concept of StegFog could be described using layers abstraction as presented in Figure 1. The layers included in the design which are utilized by each peer:

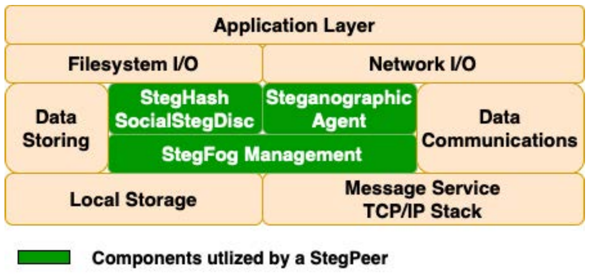


FIGURE 1. The layered architecture of StegFog.

- Application layer – an application that uses the system from the perspective of the user.
- Filesystem I/O and Network I/O - a standard system API for a filesystem and network operations
- Data communications layer - provides the interface for preparing data to be sent through networks to the other peers.
- Message service layer - it utilizes the standard TCP/IP protocol stack for any communications to be occurred between endpoints.
- Data storing layer - provides the interface for the application layer with read-write-delete operations inside the device and with managing point-to-point connection sessions; it passes commands to lower the layers.
- Local storage – represented as a non-volatile memory in the device, where data is stored.

The other layers marked as green in Figure 1 extend any peer into StegPeer with:

- StegHash/SocialStegDisc layer – the implementation of the StegHash [6] indexing method, SocialStegDisc [7] memory management and any of steganographic methods on files to hide data within the local storage [13].
- StegFog management layer – defining a set of operations on the memory sector as a group of n devices. It provides the view of the memory sector for the StegHash/SocialStegDisc layer and it manages the state of the memory sector. Furthermore, they cooperatively manage the scheme of allocation of the memory over devices, memory sectors and the whole storage system.
- Steganographic Agent – treated as a service using steganographic techniques for communications. It combines the different types of steganography such as network steganography over TCP/IP protocol stack and multimedia steganography as it is needed in StegFog.

C. SECURITY EVALUATION OF StegFog SYSTEM

The security of StegFog could be evaluated from two perspectives:

- Security of the data transmissions provided by the hidden communication layer such as TrustMAS [4] described in Section II-A. Many methods for steganography could also be applied. A summary of them is presented in [13].
- Security based on dispersion of data between devices with a logical chain created by StegHash [6].

TABLE 1. Evaluation of security requirements for StegFog.

Security criterion	If required?
Availability	YES
Usability	YES
Integrity	MIXED
Authenticity	YES
Confidentiality	YES
Ownership	MIXED

It is complemented by cyberfog security evaluation based on a well-recognized framework [21].

In StegFog, security is provided by the distribution of data and its logical connections. The resilience of the system relies on the secret of a generation function that creates the usable chain from dispersed data. When an adversary attacks the network to compromise one or a part of the devices, it would be unusable without knowledge of how to retrieve the data logically. The remaining data still could be properly read even if some parts are missing. Compromising the generation function module or capturing a device, they are very high threats, so a mechanism for triggering the automatic destruction would be taken under consideration.

This paper introduces the security analysis of StegFog by means of the cyberfog approach. It is based on the template from [21] with an extension of the classic model of the information security CIA triad (Confidentiality Integrity Availability), with three new aspects: Usability, Authenticity and Ownership. Below is a short description of the criteria of such an analysis that will be performed for the StegFog concept:

- Availability - information should be available at the request of the owner.
- Usability - information should be available in a form that allows it to be read effectively.
- Integrity - all information holders should have a consistent view of the information.
- Authenticity - the owner of the information should have access to the author of the given information.
- Confidentiality - an entity that is not the owner of the information should not have access to it.
- Ownership - the owner of the information should have access to it.

Table 1 presents the evaluation results of the analysis by assigning YES, NO or MIXED answers regarding the fulfillment of the security requirements. To summarize:

- Availability is achieved by dispersing pieces of information between end devices. Any potential attack needs to target independently operating systems. There is one known operational single point of failure (SPOF), which would lead to the unavailability of the entire system.
- Usability is realized by the introduction of an information sharing mechanism similar to the sharing of Shamir's [20] secret. Shamir's secret breakdown assumes that the information is useful - recoverable, given the k of the n message part.
- Integrity of information is not considered by cyberfog, but it could be provided by the dedicated methods in the practical implementation, such as checksums or digital signatures.
- Authenticity is provided by default in the concept. To mimic the node of the StegFog, an attacker would have to take over or modify all pieces of information that, according to the scheme of action, are distributed among the end devices. Such a scenario is highly unlikely to be realized.
- Confidentiality is realized by default in the concept and could be harden by implementation. Nodes that store pieces of data do not have access to all information. Fragments can be encrypted before the dispersing procedure, making them impossible to be read by a potentially hostile node. Notifications about a new piece of data in the system can also be encrypted. StegFog by design assumes that nodes in the network that are not instructed to store data would not be even aware of its existence among other devices.
- Ownership in cyberfog is based on the assumption that the owner of the information can request access to a given fragment at any time. In the case of compromising the node storing some information, it can be secured against interception of a given fragment using physical methods - device destruction. This is the situation that could compromise the original ownership of data in cyberfog.

In the cyberfog security approach [3], the network needs to manage a complex tradeoff between availability and confidentiality in real time depending on the users' tasks and circumstances. Evaluating the proposed design it is noticed that each mechanism of security impacts the level of complexity significantly. Table 2 compares the impact of the design on security together with operational features: memory, time and reliability. The improving effect imposed by a component on a feature is marked by $\nearrow$. The negative effect is represented by $\searrow$, i.e. higher memory consumption, slower time of operations or worse reliability. *N/D* means that there is no evidence or cause to mark them as improving nor negative effect. It is noticeable that all components impose a time penalty on the system. It is caused mainly by the fact that many mechanisms need to be executed in the order and time of computation is obviously higher. The general tradeoff of the system is to improve security in cost of memory, time and reliability.

IV. StegFog PROOF-OF-CONCEPT IMPLEMENTATION

A. INTRODUCTION TO PRACTICAL REALIZATION OF StegFog SYSTEM

In Section III the reference design of StegFog is introduced. This section presents the practical realization of the concept

TABLE 2. Evaluation of StegFog components in regard to how they are impacting system's features. The improving effect - ↗, the negative effect - ↘, *N/D* - nor improving or negative (no evidence to mark ↗ or ↘).

System component	Security	Memory	Time	Reliability
StegHash indexing scheme	↗	*N/D*	↘	*N/D*
SocialStegDisc operations scheme	↗	↗	↘	↘
Steganographic protocol agent	↗	↘	↘	↘
Memory sector	*N/D*	↘	↘	*N/D*
Storage system	*N/D*	*N/D*	↘	*N/D*
Messaging service	*N/D*	*N/D*	↘	↗

as a proof-of-concept implementation of elements of the system for empirical verification of its features. StegFog at its conceptual level is aligned with the peer-to-peer (P2P) networking approach, therefore the methods presented in the following sections could be applied by each node participating in the system. For this implementation, the protocol assumes that nodes know the identifiers of other nodes in the network. The mechanism of discovering and adding new nodes to the system are beyond this paper. Furthermore, the implementation skips the mechanism of the group secret determination, which is a random seed for the implemented generator addressing message fragments. This problem could be solved by using known protocols, such as Tree-based Group Diffie-Helman (TGDH) [22].

The key element of the implemented prototype is the addressing algorithm, which was created based on the generation of addresses proposed in the StegHash method [6]. This implementation assumes the existence of one or more spaces visible to all nodes participating in the protocol allowing the reading of the materials added to a given space. In addition, the spaces should not restrict the protocol methods and allow for the addition of new materials. Finally, the protocol does not require permission to delete or modify elements in the common space.

B. ADDRESSING MECHANISM FOR StegFog

In StegFog, the hashtags that are elements of the address in the StegHash method are replaced with alphanumeric strings. It provides that the nodes implementing the protocol can unambiguously cast the last element of the address to the node ID in the network. The common feature for the addressing in StegFog and StegHash method is the presence of a common secret, which is the random seed for the address generator. It is crucial that this secret is known only to the participants of the protocol, as this information is critical to the security of the protocol. Another common feature is the location of the address element responsible for indicating the place to look for a given piece of information. This element is at the end of each permutation. The address generation uses the mechanism of permutation counters proposed in [7], which allows for memory optimization and no need to store address dictionaries. The address generator designed for the needs of the protocol was additionally extended with the possibility of building addresses from elements that are not mapped in the node identifiers. The set of strings given to the generator input can therefore be divided into two subsets, containing:

- Significant elements - elements related to the network node identifiers.
- Insignificant elements - additional elements, not related to the protocol.

A critical element, necessary for the correct operation of the address recovery algorithm is the use of a random seed with the same value as for the generation of addresses. The seed value may change for subsequent address generation and recovery operations. The protocol does not mandate the method of data fragmentation. The requirement for correct addressing of information fragments is assigning them in the order of the generated addresses: address 1 $\rightarrow$ fragment 1, address 2 $\rightarrow$ fragment 2 $\ldots$, thus they will be correctly recreated in the reading procedure. The protocol assumes the use of steganographic channels to distribute pieces of data. The protocol in StegFog is independent of the steganographic method applied, but it requires a few elements to be stored by each steganogram:

- Data fragment.
- Address of given fragment.
- Starting address (starting permutation) needed to recover the following addresses.
- The iteration count needed to recover the next address.
- Flag coding the message type: '0' - data fragment; '1' - request a fragment of the message; '2' - notification about the availability of a new message; '3' - response to fragment request.

C. COMMUNICATION METHODS SUPPORTED BY StegFog PROOF-OF-CONCEPT

Three basic methods of the protocol are supported by the StegFog prototype: sending the message in StegFog, notifying the recipient that the message is available and reading the message from the system. Sending the message, as shown in Figure 2, can be divided into the following steps:

1) Message to be sent at a node is passed to the StegPeer module where the message is split into fragments (Frag).
2) The message fragments are placed into carriers using the selected steganographic method to create steganograms (Steg).
3) Steganograms are then addressed using strings obtained from the addressing algorithm. The iteration count and flag of the message type are also added to the steganogram.
4) Steganograms are then published in a shared space.
5) If the steganogram is addressed to the node reading it and the flag is set to '0', the node saves the data in the

local memory in the form: <fragment address, <permutation count, message fragment > >. If the steganogram is not addressed to the node it does not take any further steps.

To read the stored data, the recipient of the message should be informed about the possibility of downloading its fragments. The notification message consists of: identifier of the node to which the message is directed, a flag code of the notification message type and the address of the first fragment in the message. The notification is made available through a steganographic channel, which makes it invisible to peers using the shared space, but not participating in the StegFog. It is shown in Figure 3.

The trust between the parties is assumed by default, however, it is possible to extend it with additional encryption of the content. It could provide protection against attacks within the network and unauthorized access attempts. Reading data from StegFog is more complex than sending and notifying procedures and follows the order:

1) Request data fragment with the address received in the notification or generated by the algorithm. The content of the request consists of: data fragment address, identifier of StegPeer sending the request and the flag informing that it is a request for a fragment. The request is hidden in a steganogram and then put into the shared space where it can be retrieved by other StegPeers.
2) StegPeers fetching new data from a shared space. If a requested fragment with a given address is in the local memory of a node it makes this fragment available in the shared space. This is realized by preparing and sharing steganogram with a data fragment, a permutation counter, the identifier of the node that requested the fragment and a flag code for a response type.
3) StegPeer requesting data monitors the space and downloads any multimedia materials added there. The steganogram including the flag that indicates the response to the request and the correct identifier of the requesting node is saved in its memory. It is repeated until the value of the counter that indicates the end of the whole message by '0'. The node assembles the data fragments in the correct order.

The Reading scheme is presented in Figure 4.

D. PROTOTYPING StegFog SYSTEM

For the purpose of this research, as the building block for the platform, the BitTorrent protocol [14] was utilized, for which the operations are realized by similar nodes that share files with each other. It enables the establishment of communication in a decentralized environment of nodes easily using off-the-shelf open source software. The choice of the BitTorrent protocol is also justified due to the need to implement a steganographic channel. The basis of steganographic communication is the universality of the carriers used to create steganograms. Recently, there is an increase in the network traffic using the BitTorrent protocol [23]. The percentage of traffic is not as high as it was in 2011, where it was 52.01% for North America, but according to [23] it is was 32% for the Middle Eastern, European and African regions in 2018. It suggests that the BitTorrent protocol is recognized as a viable system to embed steganographic communication due to its prevalence. The following elements were used to implement the prototype of the steganographic node of the StegFog system and the runtime environment:

- A library with development API of BitTorrent protocol [24].
- The Java programming language, by which the StegPeer node is implemented. The compatibility with the BitTorrent library needs to be assured.
- B-Encode library [25] used to implement a custom metadata generator with.torrent extension compliant with BitTorrent protocol specification [14].
- Tracker [26] server used in the BitTorrent protocol to track torrents that exist in the network and peers (BitTorrent protocol clients).
- Containerization based on Docker to create a network for the test environment, prepare images for the tracker and peers and to make testing runtimes repeatable.
- Logging enabled for research on the protocol.
- Automating the environment using Bash scripts.

Figure 5 shows the logical view on the network where StegFog is prototyped. StegPeer can participate in the BitTorrent protocol just like other network nodes. The BitTorrent protocol relies on two networking protocols: TCP used for data communication between Peers and HTTP for communication between any Peer and the Tracker server. The Tracker server tracks Peers and torrents available on the network. On request it provides a list of Peers that have a given fragment of the message, so that Peer knows where to get the interesting fragment. The Shared Space shown previously in Figures 2, 3 and 4 is represented in BitTorrent by:

- Utilizing files with the description of the torrent with the extension.torrent, which are often hosted by other websites.
- Utilizing a memory space in devices that have downloaded a given fragment of the torrent by making them available to the other nodes.

The BitTorrent uses the underneath of the Kademila protocol [27] for its operation. Existing research, such as [27], suggests how to harden communication based on the Kademila protocol by optimizing the protocol parameters. The resilience of the network could be increased by the proper setup of a bucket size k-bucket, where routing information to other network nodes is stored. The tracker server could be recognized as a single-point-of-failure, however it is possible to run the system without a tracking server by including the BitTorrent DHT protocol controller [28]. Operating without a tracking server is aligned with the cyberfog concept (Section II-B). Researching this aspect is beyond this paper and it could be investigated in the future. Still, using the

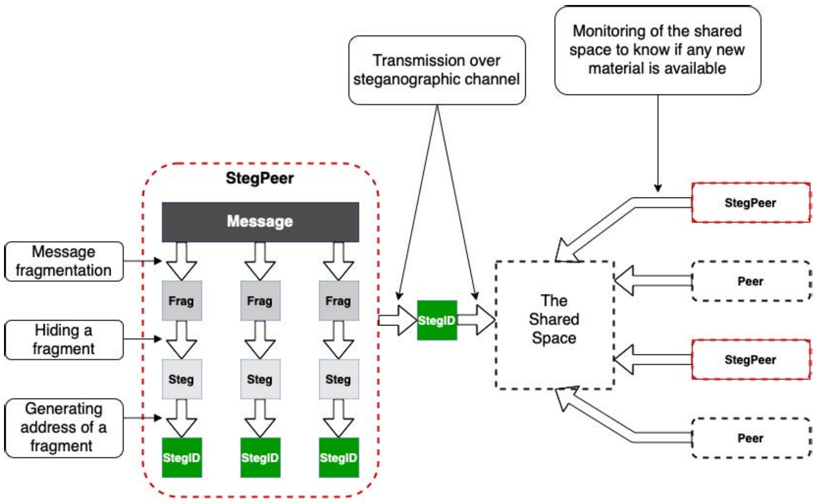

Transmission over steganographic channel
Monitoring of the shared space to know if any new material is available
StegPeer
Message
Message fragmentation
Hiding a fragment
Generating address of a fragment
Frag
Steg
StegID
The Shared Space
StegPeer
Peer
StegPeer
Peer

FIGURE 2. StegFog communication methods: Sending.

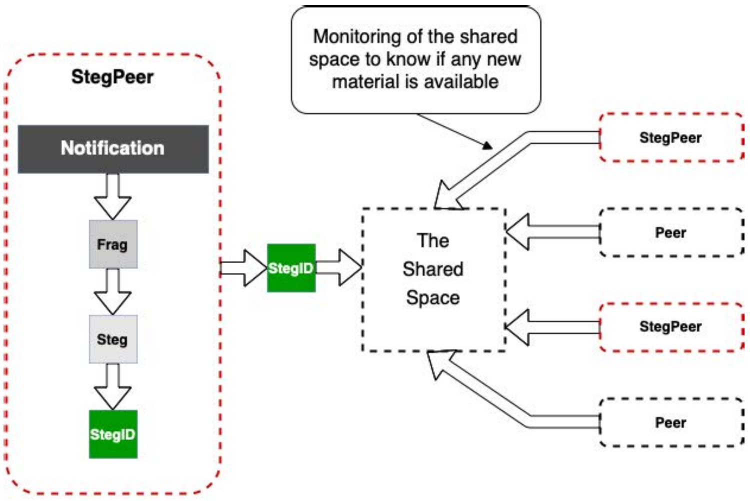


FIGURE 3. StegFog communication methods: Notification.

Tracker server is closer to real networking environments where the BitTorrent protocol is used.

V. EXPERIMENTS

A. SIMULATION ENVIRONMENT

To conduct research on the implemented prototype of StegFog, three simulation networks were deployed consisting of four, six or eight nodes together with the tracker node (Tracker). The tests were performed in the cloud environment as a virtual machine with Ubuntu 16.04 installed. The instance had 60 GB of RAM and a 16-core 6th generation Intel Xeon CPU. The software modules were managed by Docker and its relevant configuration files. The first step to prepare the simulation environment was to create subnets for future containers. In the next step, the tracker docker image was created based on the public system image Ubuntu 16.04 and the BitTorrent tracker library [26]. The container, which was created on the basis of the tracker image, was configured to enable outside communications and to monitor the current statistics, such as the number of connected peers, their type and the version of the BitTorrent protocol client. The image of the communication node was created with Ubuntu 16.04 and the correct Java runtime environment. TheStegPeer image exposes ports 6891 and 49001 for the BitTorrent protocol library [24]. It also creates the required folder structure to copy the carriers (multimedia files) to be used by StegFog to the runtime of a container. Alternatively, the StegPeer container provides logs and torrent files, which can be accessed from the hosting system through a web portal or command line interface.

B. EVALUATION OF StegFog PROOF-OF-CONCEPT IMPLEMENTATION

1) ADDRESSING SCHEME GENERATOR ANALYSIS

The address generator described in Section IV-B differs from a simple hash function for a given input string by combining significant elements to identify a steganographic node with insignificant elements of pseudorandom characters.

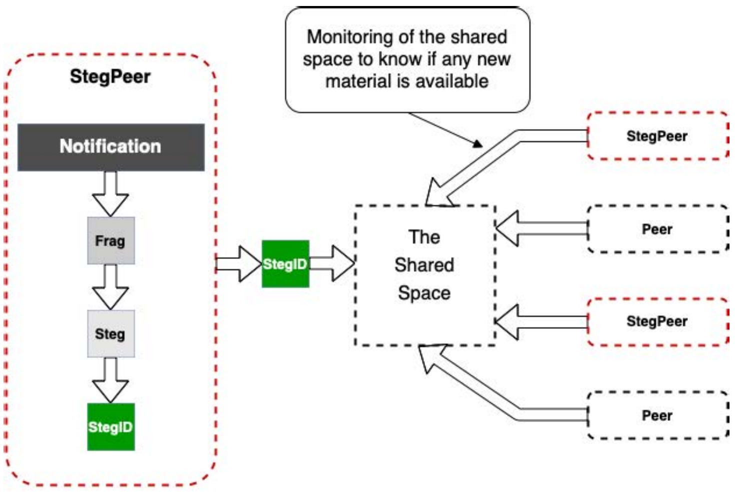


FIGURE 4. StegFog communication methods: Reading.

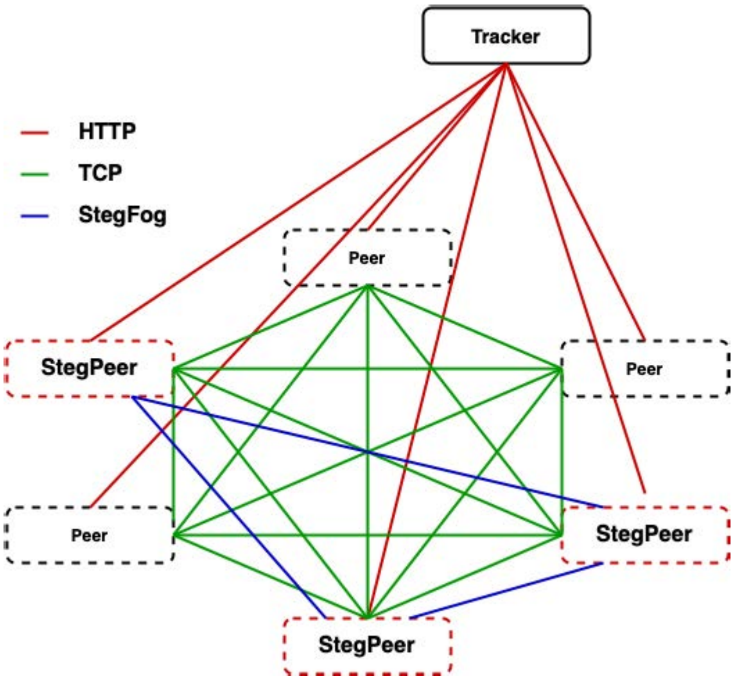


FIGURE 5. Logical view on the network with StegFog communication applied.

Referring to the concept presented in Section III to correctly address a given fragment of a message, the identification substring should be placed at the end of the set given at the input of the generator. This procedure limits the space of possible combinations of the input set. The maximum number of usable addresses can be calculated with the following equation:

$$P = k * (n - 1)! \tag{2}$$

where:

- P - the maximum possible number of permutations to generate addresses.
- n – size of the input set to be permuted (sum of significant and insignificant elements).
- k – the number of significant elements in the set.

To investigate the impact of such a procedure used in the address generator the probability distributions of characters appearing in the addresses were analyzed. The hash function was used to construct the reference distribution, which generates permutations of the input set in a pseudo-random manner, without repetitions. This function takes as a parameter the number of addresses needed to be generated and a set of elements. During the experiment, the probabilistic algorithm implemented in the generator and the hash function used the same random seed value. The result of the address generator and hash function is a list of permutations of the elements for the input file in the following format:

```
iB, Hu, im, Xt, QB, lI, xQ, Hz
```

Relative entropy, also known as the Kullback-Leibler divergence [29], was chosen as the measure determining the discrepancy of the probability distributions:

$$d_{KL}(P, Q) = \sum_{i} p(i) \log_2 \frac{p(i)}{q(i)} \tag{3}$$

where:

- d - relative entropy of the distributions of discrete random variables P and Q.
- p - probability of the sample value of a random variable P.
- q - probability of the sample value of a random variable Q.

This measure is a classic element for information theory with practical considerations in cybersecurity. An example of the use of this measure in cybersecurity is the detection of automatically generated domains by the so-called algorithms DGA (Domain Generation Algorithms) [30], [31], and [32]. These algorithms are implemented in the malware to create C2 channels between infected machines and the control server. Examples of this type of software are Skybot and Styx [33]. To find anomalies in strings, e.g., domain names, file names or processes, entropy is also used. In [34], this approach was used to detect anomalies in the domain names. Furthermore, it was shown by [34] that the relative entropy allows the detection of the differences between the tested samples more clearly when compared to the classical entropy.

The set consisting of the addresses obtained from the generator and the hash function was modeled using the n-gram model [35]. This model is used in cybersecurity, for example, to analyze executable files for the presence of malware [36]. N-grams are also used in other fields such as biology [37] and computational linguistics [38]. In this study, in addition to understanding the impact of the disturbance in the probability distribution introduced by the generator based on the StegHash [6] method, four other questions are raised:

- Question 1: How does the size of n in the n-gram model affect the presence of the disturbance?
- Question 2: What is the influence of the size of the examined address space on the presence of the disturbance?
- Question 3: How does a change in n in the n-gram model and address space size affect the entropy of distributions?
- Question 4: Does changing the ratio of identifying elements to insignificant elements affect the relative entropy of the address distributions as defined by Eq. 3?

Converting the set of addresses to the form n-grams consists of concatenating all addresses in the set into one string and creating the appropriate length n-grams. In the examined case, unigrams (1-gram), bi-grams (2-gram) and tri-grams (3-gram) were generated. The n-gram distribution is discrete, therefore the probabilities of n-gram occurrences in the set are calculated based on the number of occurrences of the given n-gram. Before calculating the relative entropy, the previously calculated n-grams probability vectors from the different distributions must be standardized. The standardization aligns the lengths of the probability vectors. This is realized by adding the appropriate number of zero probabilities n-grams, which previously did not appear in a given distribution. However, these distributions still need to be smoothed out to eliminate the zero-frequency problem [38]. For n-grams derived from the second distribution, a very small, non-zero, fixed-value probability is assigned, and all other probability values from the original distribution are reduced by that fixed amount so that the sum of the vector probabilities is still 1. The results of the measurements: entropy of addresses from the generator, entropy of addresses from the hash function (reference permutation) and relative entropy of these distributions, are presented in the Figures 6, 7 and 8.

The samples are from the address space range 10 through 20110, in increments of 200. During the measurements, the system was configured with the following parameter values: the number of significant elements is four and the size of the set to be permuted is eight. According to Eq. 2, the maximum number of addresses that can be generated in this configuration is 20,160.

2) HOW DOES THE SIZE OF N IN THE N-GRAM MODEL AFFECT THE PRESENCE OF THE DISTURBANCE?

Comparing the results of relative entropy measurements from Figure 6, it can be concluded that the selection of the n-gram model has an impact on the visibility of the disturbance introduced by the address generator. In the case of the unigram model, the relative entropy of the distributions is zero, while for the bi-gram and tri-gram it is positive and answers Question 1.

3) WHAT IS THE INFLUENCE OF THE SIZE OF THE EXAMINED ADDRESS SPACE ON THE PRESENCE OF THE DISTURBANCE?

Based on the results presented by Figures 6, 7 and 8, it is seen that the disturbance introduced by the generator is most visible for the small size of the compared address space, especially below 100 addresses. This behavior is presented in Figure 6 by higher values of the relative entropy and in Figures 7 and 8 by larger differences in the entropy values from two different distributions, compared to larger portions of the address space. This observation answers Question 2.

4) HOW DOES A CHANGE IN N IN THE N-GRAM MODEL AND ADDRESS SPACE SIZE AFFECT THE ENTROPY OF DISTRIBUTIONS?

Figures 7 and 8 show that the size of the investigated address space has no influence on the entropy values. Based on the entropy of the distribution from the generator (Figure 7) it can be seen that the values from this distribution do not show an upward or downward trend. Referring to Question 3, the entropy value increases depending on the selection of the n-gram model, which results from the structure of n-grams. Along with the increase in the parameter n, the number of characters in n-grams is increasing, which, if the characters are different, allows you to store more information in them.

Another observation seen for the bi-grams in Figure 7 and tri-grams in Figure 8 is the fact that in most measurements

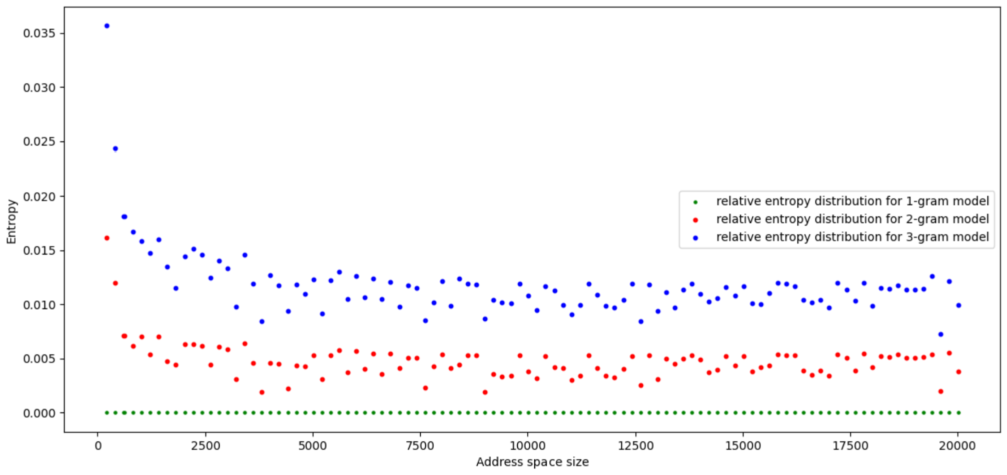


FIGURE 6. **Comparison of relatives entropies based on 1-gram, 2-gram and 3-gram models of strings from the generator and from the hashing function.**

the entropy values from the address generator distribution are smaller than those from the hash function. In about one hundred samples visible in the discussed graphs, only in two cases the value of entropy coming from the generator is higher than that of the hash function - in the bi-gram model and only once in the tri-gram model. For the unigram model, n-grams reach the same values, therefore the influence of the generator is not visible.

5) DOES CHANGING THE RATIO OF IDENTIFYING ELEMENTS TO INSIGNIFICANT ELEMENTS AFFECT THE RELATIVE ENTROPY OF THE ADDRESS DISTRIBUTIONS?

To answer Question 4 and examine the influence of the ratio of significant elements to non-significant elements, the following system parameters were setup: length of permutation is 24 and size of set to be permuted is 16. These parameters cause the address space to increase to 159,667,200. The number of significant elements remained the same as four, and the number of insignificant elements was doubled to eight. The bi-gram model will be used to investigate the difference in relative entropy, and the samples will be in the range from 10 to 20110, with a step of 200. Based on Figure 9 in the analyzed address space it can be observed that in the vast majority of this space, the relative entropy with the hash function reaches lower values for the system settings with the ratio of significant to insignificant elements equal to 1:2 in comparison to the 1:1 setting. Thus, by increasing the number of insignificant elements, we reduce the difference in the distribution of addresses from the generator, with the distribution coming from the hash function. The mean value of the relative entropy for a 1:2 ratio of significant to insignificant is 1.6 times smaller than for a 1:1 ratio, which answers Question 4.

6) OPERATIONAL MEASUREMENTS WITHIN StegFog PROOF-OF-CONCEPT SYSTEM

During research on the proposed protocol, a series of measurements was performed to determine its properties depending on the number of nodes in the network and the size of the transferred data. Additionally, an attempt was made to determine the steganographic bit rate of the implemented system prototype. Two test scenarios were designed to test the timing of the protocol. During the measurements, three phases in the protocol were distinguished: sending, notification, reading.

The first scenario consists of sending a message 102 characters long distributed among five steganograms. This test was performed for each of the simulation configurations with 4, 6 and 8 nodes (Section V-A). The testing message was sent between two selected nodes. It was repeated three times for each configuration. The time results for each of the phases are presented in Table 3 in the form of an arithmetic mean with the variance.

Based on the results presented in Table 3, it can be concluded that for the same message size, the duration of the loading and reading phases may differ. The time of the notification phase is constant, regardless of the size of the network. For each steganogram containing a message fragment, a unique address in the network is generated. Table 4 shows the addreeses generated per each steganogram for three configurations.

The operation of the address generating function depends on the list of addresses available in the network. The node address is encoded in two characters, so the last two characters of the fragment address determines the node address that is to store the fragment. Comparing the addressing for three different network sizes in Table 4, it can be seen that the number of fragments sent between selected nodes differs depending on the configuration. The same situation occurs when the message fragments are downloaded by the second node during the Reading phase. It comes from one of two situations:

- A node requests a fragment that is stored locally on that node.
- A node wants to store a fragment and the protocol selects its own memory space to do so.

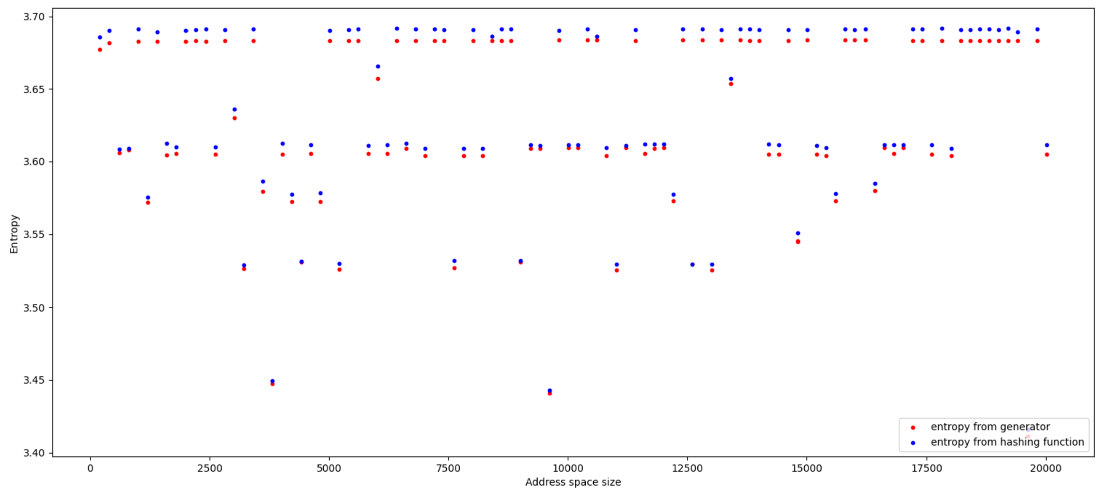


FIGURE 7. Comparison of entropies between 2-gram models of strings from the generator and from the hashing function.

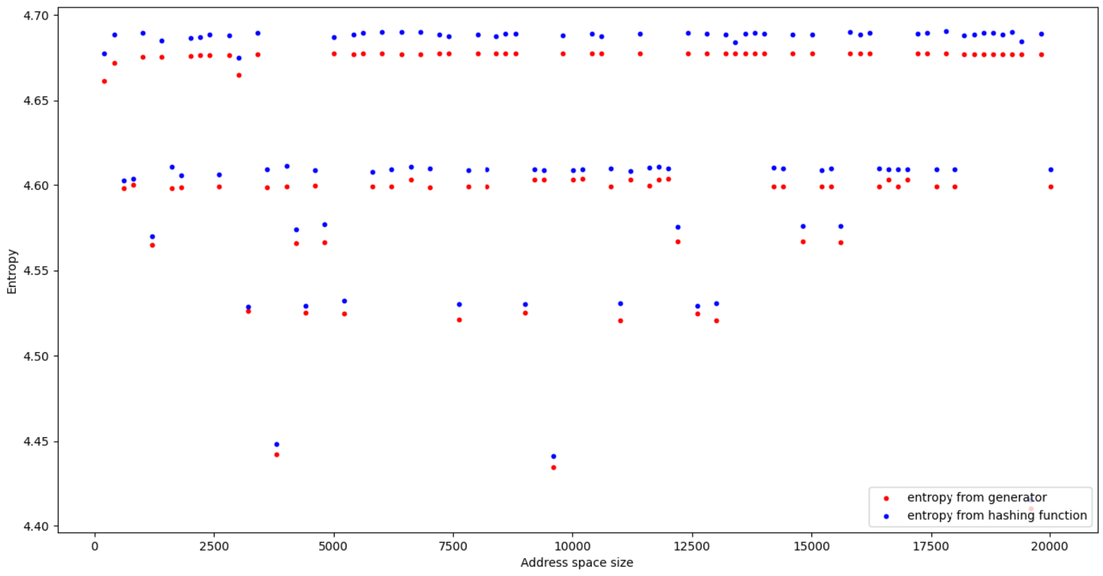


FIGURE 8. Comparison of entropies between 3-gram models of strings from the generator and from the hashing function.

TABLE 3. Time measurement results in Scenario 1 of StegFog operations. (*T* - time, *Var* - variance).

Protocol phase	4 nodes		6 nodes		8 nodes	
	T [s]	Var [s]	T [s]	Var [s]	T [s]	Var [s]
Sending	187	2	307	0	307	1
Notification	16	1	17	0	16	2
Reading	383	6	303	27	370	54

These time differences were not visible during the Notification phase, because in this case the information is always hidden in one steganogram. Taking into account these observations, Table 5 was prepared showing the time dependencies between the shared and downloaded steganograms and the network size. The unit of values in Table 5 is S [s / 1 steg], which means the number of seconds needed to transfer one steganogram in the tested configuration of nodes.

The difference in the values between the Sending and Reading phases is that:

- During sending the node makes the message fragments available for download via *seeding* mechanism of BitTorrent [14],
- Reading of each fragment consists of sending the request to provide a fragment and the fetching of the fragment from the node containing it in the *leeching* mode [14].

As can be seen in Table 4, along with the increase in the number of nodes in the network, the number of characters addressing the fragments has increased. Based on results showing the average times for sharing and downloading the steganogram (Table 5), it can be concluded that the transfer of the steganogram in the network of nodes implementing the StegFog concept is independent of the number of nodes in the network, as well as the impact of the addressing size being negligible.

In the second scenario, the important parameter is that the message length embed into a steganogram is set to 25.

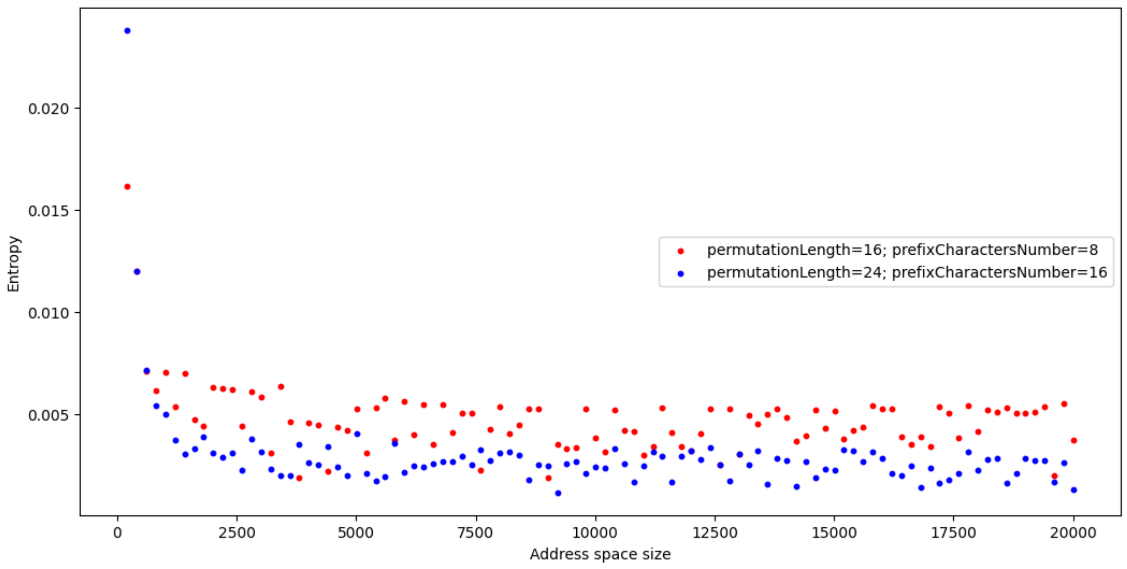


FIGURE 9. Comparison of relatives entropies based on 2-gram models of strings from the generator and from the hashing function with the extended set of insignificant elements in the address string (configured by `prefixCharactersNumber` parameter).

TABLE 4. Addresses of steganograms generated in StegFog performace testing scenarios.

Steganogram No	4 nodes
1	D4B2dUA1p9wsGVC3
2	D4dUC3B2GVp9wsA1
3	A1wsdUGVp9B2D4C3
4	p9B2wsA1dUC3GVD4
5	C3dUB2D4p9wsGVA1
	6 nodes
1	qYF63sD4ypA1jWB2E5C3
2	B2A1qYD4E5C33sjWypF6
3	jWA1F6qYC3B23sypD4E5
4	qY3sA1F6D4jWypE5B2C3
5	3sE5D4A1jWqYypF6C3B2
	8 nodes
1	G7H8uTE50qpXafB2A1D4C3F6
2	pXA1E5D4H8F6B20qafuTG7C3
3	E5H8D4A10qafC3pXF6uTB2G7
4	C3F6A1uTE5G70qH8afB2pXD4
5	pXH8D4E5C3uTG70qafA1B2F6

Three messages with lengths of 50, 100 and 200 characters were prepared for sending between two selected nodes. Such communication means fragmentation into 2, 4 and 8 parts, respectively. Observing the results presented by Table 6 with the corresponding plot in Figure 10, a linear increase in the time for sending and reading messages depending on the number of steganograms sent is noticed. Additionally, it overlaps the measurements of the average sending and reading times from Table 6, to check the computational overhead in comparison with the five message fragments. It means that the time differences introduced by the computation are negligible.

7) ESTIMATING STEGANOGRAPHIC RATE OF StegFog SYSTEM

It is not possible to accurately determine the steganographic rate of StegFog as it does not need to be implemented using any specific steganographic methods for local data storage on StegPeers and communications between them. The general formula for the steganographic flow in a given implementation can be expressed by the formula:

TABLE 5. Measurement results in Scenario 1 of StegFog operations, *S* as [s / 1 steg].

Protocol phase	4 nodes S [s / 1 steg]	6 nodes S [s / 1 steg]	8 nodes S [s / 1 steg]
Sending	62	62	61
Notification	16	17	16
Reading	38	38	37

$$S = \frac{c}{v} \tag{4}$$

where S is the steganographic bit rate expressed in [B / s], c is the carrier capacity expressed in [B / 1 steg] - the number of bytes that can be hidden in the carrier and v is the number of seconds needed to transmit one steganogram expressed in [s / 1 steg].

The well-known and easy to implement Least Significant Bit (LSB) steganography method was applied to hide data in PNG images with some modifications to extend the number of parameters transmitted with the data. For the purpose of this research, LSB image steganography method is considered as the reference application of a steganographic method within the StegHash/SocialStegDisc layer of StegFog system, especially to streamline the proof-of-concept implementation, experiments and performance estimations. Any other methods such as text and multimedia steganography [13], or more advanced algorithms e.g. [39] or [18] could be used on that layer shown in Figure 1. The concept as [18] could contribute to improve the security of StegFog by adding another distributed approach focusing on multimedia steganography. It could add more security on the lower layer

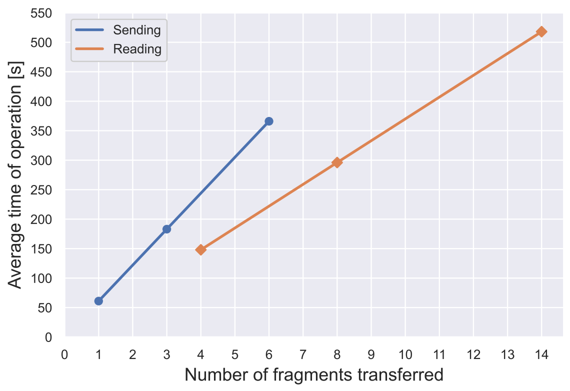


FIGURE 10. Plot of the linear dependence between number of fragments transferred during (Sending; Reading) operations and time to complete each one. Based on Table 6.

of the system as complement to the higher layer mostly developed by this paper.

Two types of carriers were used during the measurements: images in PNG format with a resolution of 1024 x 1024 and 2028 x 1521 pixels. When generating the steganogram, the carriers were collected in a pseudo-random manner, so it was assumed that the same amount was used during the research. The metadata in the steganogram has a size of 39 B. The implementation of the LSB method encodes one bit per pixel. By averaging the number of pixels available in carriers and the size of the metedata, the average capacity of the carriers used was about 26KB.

The data from Section V-B6 was used to calculate the average time in seconds needed for the transmission of one steganogram. The sum of the average times needed to perform the measurements presented in Table 3, and the total number of transferred steganograms in all protocol phases were taken into account. The total time was 1906 seconds, and 44 steganograms were transferred during it, which gives 43 [s / 1 steg]. Putting the obtained results for c and v into the Eq. 4, the steganographic bit rate is 605 [B / s].

C. DISCUSSION AND EVALUATION

This paper analyzes the information security of the cyberfog concept. For verification, an extended template was used, taken from the literature [21], which, in addition to the basic security aspects, such as confidentiality, integrity and availability, also takes into account: usefulness, authenticity and ownership of information. The obtained results show that this method does not fully protect against attacks on data integrity and ownership. Potential extensions of the method as proposed in Section IV-D may allow all security aspects to be met in the final implementation.

For the needs of the research, a distributed steganographic system was implemented, allowing the usability of the protocol prototype to be checked and its basic functions to be examined. However, this is not the final solution. As mentioned in Section IV-D, the next version of the concept should be expanded by including a node discovery mechanism, sharing secrets for the address generator seed, as well as developing a link reservation method or enhancing the library [24] to ensure full asynchronous steganographic communication. The protocol features related to the address space size look promising, the maximum value of which can be calculated using the Eq. 2. For the size of the element set input to the address generator n, the size of the space increases (n-1)! times. It is worth noting that the number of signifiers reflecting the protocol node identifiers does not have a significant influence on the size of the space. To obtain a larger space size, the ratio of insignificant to significant elements should be increased.

The results of the relative entropy studies of the probability distributions from the generator and the pseudorandom hash function presented in Section V-B1 do not show any significant disturbance from the generator. For the 1-gram model, they are not recognized at all, and for the 2-gram and 3-gram models, respectively higher, but not exceeding 0.08. For 98% of the samples, the relative entropy of the distributions is below 0.025. From the graphs presented in Section V-B1, it can be seen that the entropy from the generator is in most cases slightly lower than that obtained from the samples obtained with the hash function. When looking at these cases, it is worth remembering that in a real scenario, there is a small probability that the observer would have a perfectly separated set of addresses coming only from steganographic communication. In further research, the degree of undetectability will be assed in the context of addressing depending on its number among other strings of characters indexing multimedia materials in a given system, in which the protocol was embedded. Section V-B6 presents the timing results of the StegFog operations. Due to the implementation of the prototype, the reading method takes the longest time. For each of the pieces of information not stored on the requesting device, it is necessary to provide a steganogram containing the request for a given fragment, and then retrieve it from the node that owns it. This behavior is determined by the BitTorrent protocol specification. It was empirically verified that the network size for the tested configurations did not affect the average time of sending and reading the steganograms. The results of research related to the time of sending and reading information from the system show a linear increase depending on the number of steganograms involved in the communication. The values shown on Figure 10, are consistent with the average information transfer times for each protocol method obtained in this study. The difference in the average time of sending and reading is related to the protocol implementation. In the reading phase, the sending of the fragment from the node with it to

TABLE 6. Summary of the measurements of StegFog operations in Scenario 2.

Aspect	2 fragments	4 fragments	8 fragments
Sending	63 s	186 s	373 s
Sending (avg.)	61 s	183 s	366 s
Notification	17 s	14 s	15 s
Reading	149 s	306 s	528 s
Reading (avg.)	148 s	296 s	518 s
Fragments sent	1	3	6
Fragments read	4	8	14

the requesting node consists of two steganograms. The first one uses the seeding mechanism of the BitTorrent protocol, and the second one downloads the material in the leaching mode. The two modes differ in their average durations, with the seeding mode being longer. The sending phase uses only the longer mode, while the reading phase uses both, which lowers the average steganogram transmission time.

For the implementation of the protocol using BitTorrent, the steganographic bit rate was calculated at 605 [B / s]. However, this value should not be treated as the maximum bit rate achievable by the protocol as it is strongly dependent on the capacity of the carriers and the chosen method of multimedia steganography, which is independent of the protocol concept defined in Section III. The above results and the prototype of the protocol can form the basis for further work bringing it closer to solving real problems, such as storing sensitive data and ensuring appropriate access to it. This example shows that steganography is used not only in malware communication, but can also protect data and privacy of citizens against abuse by the state and other institutions.

VI. SUMMARY

As part of the work, a prototype of a distributed steganographic system implementing a protocol dedicated for a new concept in the field of cybersecurity, and Cyberfog was realized. The conducted research and analysis of the protocol may constitute the basis for determining the manner and scope of the method's use in view of the problems related to the storage and sharing of sensitive information mentioned in the introduction. The results and observations from the conducted information security analysis in Section III-C can be helpful for the further development of the cyberfog method, thus supplementing it with security aspects, such as integrity or ownership. The prototype of the proposed protocol uses the BitTorrent network as a P2P platform offering multimedia sharing in a decentralized environment. The nature of this type of network may make it difficult to detect C2 channels between the managing server and devices in the botnet. The paper presented the simulation environment and the description of the research carried out on the protocol. The address generator has been analyzed. The formula for the size of the address space has been given depending on the parameters given at the generator input. Using the relative entropy and the n-gram model, the disturbance introduced by the generator was examined in comparison to the result obtained from the hash function. How the system parameters related to the generator affect the size of the disturbance as assessed. Additionally, it was examined how the selection of the language model - n-gram, influences the detection of the disturbance. The paper includes research on protocol timing. As part of this bit of the work, the following were verified:

- Influence of network size on execution times of protocol methods with constant information size.
- Relationship between the duration of sending and reading information, and the size of the information.
- Steganographic bandwidth achieved in the simulation environment.

The implemented example of the system can serve as an inspiration for entities dealing with detection and counteracting malicious software and revealing vulnerabilities in websites using the BitTorrent protocol to place steganographic channels in them. In future research, some aspects could be considered:

- In the case of using the protocol in networks with a variable number of nodes, it is worth considering extending it with the method of discovery or registration of new steganographic nodes.
- When developing a system based on the protocol defined in Section III, whose security relies on the secrecy of the address generator, it is worth considering a mechanism for finding a secret or its cyclic change.
- A further stage for research on the address generator based on the adapted StegHash method may be to check the distribution difference between the addresses created by the generator and the selected set of torrent names presented on a given website. The result of this test would be to check the discoverability of the addresses that are torrent names among other torrent names, and not from the address generator.

REFERENCES

[1] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[2] S. Sicari, A. Rizzardi, and A. Coen-Porisini, "Insights into security and privacy towards fog computing evolution," *Comput. Secur.*, vol. 120, Sep. 2022, Art. no. 102822.

[3] A. Kott, A. Swami, and B. West, "The fog of war in cyberspace," *Computer*, vol. 49, no. 11, pp. 84–87, 2016. [Online]. Available: https://www.computer.org/csdl/magazine/co/2016/11/mco2016110084/13rRUEgs2PJ

[4] K. Szczypiorski, I. Margasiński, W. Mazurczyk, K. Cabaj, and P. Radziszewski, "TrustMAS: Trusted communication platform for multi-agent systems," in *On the Move to Meaningful Internet Systems: OTM 2008*. Berlin, Germany: Springer, 2008, pp. 1019–1035.

[5] M. Chapman, G. I. Davida, and M. Rennhard, "A practical and effective approach to large-scale automated linguistic steganography," in *Information Security*, G. I. Davida and Y. Frankel, Eds. Berlin, Germany: Springer, 2001, pp. 156–165.

[6] K. Szczypiorski, "StegHash: New method for information hiding in open social networks," *Int. J. Electron. Telecommun.*, vol. 62, no. 4, pp. 347–352, Dec. 2016. [Online]. Available: http://journals.pan.pl/Content/101655/PDF/48.pdf

[7] J. Bieniasz and K. Szczypiorski, "SocialStegDisc: Application of steganography in social networks to create a file system," in *Proc. 3rd Int. Conf. Frontiers Signal Process. (ICFSP)*, Sep. 2017, pp. 76–80.

[8] J. Bieniasz and K. Szczypiorski, "Methods for information hiding in open social networks," *J. Univ. Comput. Sci.*, vol. 25, no. 2, pp. 74–97, 2019.

[9] J. Bieniasz and K. Szczypiorski, "Towards empowering cyber attack resiliency using steganography," in *Proc. 4th Int. Conf. Frontiers Signal Process. (ICFSP)*, Sep. 2018, pp. 24–28.

[10] X. Liao, Q.-Y. Wen, and S. Shi, "Distributed steganography," in *Proc. 7th Int. Conf. Intell. Inf. Hiding Multimedia Signal Process.*, Oct. 2011, pp. 153–156.

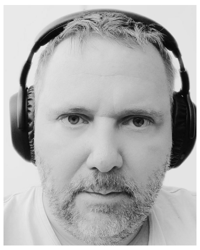
[11] N. Fazio, A. Nicolosi, and I. Perera, "Broadcast steganography," in *Topics in Cryptology—CT-RSA 2014*. Cham, Switzerland: Springer, 2014, pp. 64–84. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-04852-9_4

[12] A. Alston, "A steganographic design paradigm for general steganographic objectives," Columbia Univ., Steganography Project Rep., Spring 2017, pp. 1–30. Accessed: May 30, 2022.

[13] W. Mazurczyk, S. Wendzel, S. Zander, A. Houmansadr, and K. Szczypiorski, *Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and Countermeasures*, 1st ed. Hoboken, NJ, USA: Wiley, 2016.
[14] BitTorrent. (2008). *BEP 3: The BitTorrent Protocol Specification*. Accessed: May 30, 2022. [Online]. Available: https://www.bittorrent.org/beps/bep_0003.html
[15] A. Venčkauskas, N. Morkevičus, G. Petraitis, and J. Čeponis, "Covert channel for cluster-based file systems using multiple cover files," *Inf. Technol. Control*, vol. 42, no. 3, pp. 260–267, Sep. 2013.
[16] L. Moyou Metcheka and R. Ndoundam, "Distributed data hiding in multi-cloud storage environment," *J. Cloud Comput.*, vol. 9, no. 1, pp. 1–15, Dec. 2020.
[17] S. W. M. Tcheunteu, L. M. Metcheka, and R. Ndoundam, "Distributed data hiding in a single cloud storage environment," *J. Cloud Comput.*, vol. 10, no. 1, pp. 1–15, Aug. 2021.
[18] X. Liao, J. Yin, M. Chen, and Z. Qin, "Adaptive payload distribution in multiple images steganography based on image texture features," *IEEE Trans. Depend. Sec. Comput.*, vol. 19, no. 2, pp. 897–911, Apr. 2022.
[19] E. Griffor, C. Greer, D. Wollman, and M. Burns, "Framework for cyber-physical systems: Overview," NIST-Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. NIST SP 1500-201, Jun. 2017, vol. 1. [Online]. Available: https://www.nist.gov/publications/framework-cyber-physical-systems-volume-1-overview
[20] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979, doi: 10.1145/359168.359176.
[21] D. B. Parker, *Toward a New Framework for Information Security?* Hoboken, NJ, USA: Wiley, 2012, ch. 3, pp. 3.1–3.23. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118851678.ch3
[22] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement," *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 1, pp. 60–96, Feb. 2004.
[23] A. Sachdeva. (2018). *BitTorrent Popularity and Online Piracy is Increasing Again: Here's Why*. Accessed: May 30, 2022. [Online]. Available: https://fossbytes.com/bittorrent-popularity-online-piracy-is-increasing-again-netflix/
[24] *BT: A Full-Featured BitTorrent Implementation in Java 8*. Accessed: May 30, 2022. [Online]. Available: https://github.com/atomashpolskiy/bt
[25] *B-Encode: Library to Encode and Decode B-Encoded Documents*. Accessed: May 30, 2022. [Online]. Available: https://github.com/adaxi/Bencode
[26] *BitTorrent-Tracker: Simple, Robust, Bittorrent Tracker (Client & Server) Implementation*. Accessed: May 30, 2022. [Online]. Available: https://github.com/webtorrent/bittorrent-tracker
[27] H. Heck, O. Kieselmann, and A. Wacker, "Evaluating connection resilience for the overlay network Kademlia," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 2581–2584.
[28] BitTorrent. (2020). *BEP 5: DHT Protocol*. Accessed: May 30, 2022. [Online]. Available: https://www.bittorrent.org/beps/bep_0005.html
[29] T. Cover and J. Thomas, *Elements of Information Theory*. Hoboken, NJ, USA: Wiley, 2012.
[30] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, "Detecting algorithmically generated malicious domain names," in *Proc. 10th Annu. Conf. Internet Meas. (IMC)*. New York, NY, USA: Association for Computing Machinery, 2010, pp. 48–61, doi: 10.1145/1879141.1879148.
[31] T. Barabosch, A. Wichmann, F. Leder, and E. Gerhards-Padilla, "Automatic extraction of domain name generation algorithms from current malware," in *Proc. NATO Symp. IST-111 Inf. Assurance Cyber Defense*, Koblenz, Germany, 2012, pp. 2-1–2-14.
[32] MITRE ATT&CK. (2020). *T1568.002: Dynamic Resolution: Domain Generation Algorithms*. Accessed: May 30, 2022. [Online]. Available: https://attack.mitre.org/techniques/T1568/002/
[33] SANS Internet Storm Center. (2015). *Detecting Random—Finding Algorithmically Chosen DNS Names (DGA)*. Accessed: May 30, 2022. [Online]. Available: https://isc.sans.edu/forums/diary/Detecting+Random+Finding+Algorithmically+chosen+DNS+names+DGA/19893/
[34] Red Canary. (2018). *Using Entropy in Threat Hunting: A Mathematical Search for the Unknown*. Accessed: May 30, 2022. [Online]. Available: https://redcanary.com/blog/threat-hunting-entropy/
[35] W. B. Cavnar and J. M. Trenkle, "N-gram-based text categorization," in *Proc. 3rd Annu. Symp. Document Anal. Inf. Retr. (SDAIR)*, 1994, pp. 161–175.
[36] R. Zak, E. Raff, and C. Nicholas, "What can N-grams learn for malware detection?" in *Proc. 12th Int. Conf. Malicious Unwanted Softw. (MALWARE)*, Oct. 2017, pp. 109–118.
[37] Z. Volkovich, V. Kirzhner, A. Bolshoy, E. Nevo, and A. Korol, "The method of N-grams in large-scale clustering of DNA texts," *Pattern Recognit.*, vol. 38, no. 11, pp. 1902–1912, Nov. 2005.
[38] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2009.
[39] X. Liao, Y. Yu, B. Li, Z. Li, and Z. Qin, "A new payload partition strategy in color image steganography," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 3, pp. 685–696, Mar. 2020.

JĘDRZEJ BIENIASZ was born in Warsaw, Poland. He received the M.Sc. degree (Hons.) in telecommunications from the Faculty of Electronics and Information Technology (FEIT), Warsaw University of Technology (WUT), in 2017. He was involved in establishing B.Sc. studies under cybersecurity program at WUT, in 2019, where he is teaching various topics in cybersecurity (introduction courses, network security, application security, digital forensic, and incident response). He is currently an Assistant Professor with the Cybersecurity Division, FEIT, Institute of Telecommunications (IT), WUT, where he is also the Head and the Co-Founder of the Cybersecurity Center. He is the author or the coauthor of over 20 papers. He has various experience from projects with academia and industry around cybersecurity, IT, and networking. His research interests include cybersecurity data science and analytics, cyber threat modeling and hunting, secure systems, applications, and architectures.

PATRYK BĄK was born in Pionki, Poland. He received the B.Sc. and M.Sc. degrees in telecommunications from the Faculty of Electronics and Information Technology (FEIT), Warsaw University of Technology (WUT), Poland, in 2018 and 2020, respectively. He started professional career in IT industry, in 2017. In 2019, he was promoted to the Technical Project Leader and he lead the development of multiple projects for banking industry. He is currently building DevOps platforms in on-premises data centers and public clouds. He is a Certified Google Cloud Engineer. He is the author or coauthor of two papers.



KRZYSZTOF SZCZYPIORSKI (Senior Member, IEEE) was born in Warsaw, Poland. He received the M.Sc. (Hons.), Ph.D. (Hons.), and D.Sc. (Habilitation) degrees in telecommunications from the Faculty of Electronics and Information Technology (FEIT), Warsaw University of Technology (WUT), Warsaw, in 1997, 2007, and 2012, respectively. He also completed his postgraduate studies in psychology of motivation at the University of Social Sciences and Humanities, Warsaw, in 2013, and the Master of Business Administration (M.B.A.) degree from the Warsaw University of Technology Business School, in 2022. He graduated in advanced networking from Budapest Tech (now Óbuda University), Hungary, in 2003, and the Hass School of Business, University of California at Berkeley, in 2013. Since 2015, he has been the Co-Founder and the Research and Development Director of Cryptomage S.A., a cybersecurity company. He has been the Head and the Co-Founder of the Research Centre for Cybersecurity and Data Science, WUT, since 2020, and the B.Sc., M.Sc., and M.B.A. programs in cybersecurity at WUT, since 2019. He is currently a Professor of telecommunications with the Institute of Telecommunications (IT), FEIT, WUT, where he is also the Head and the Co-Founder of the Cybersecurity Division. He is the author or the coauthor of over 220 papers, three patent applications (one is granted), and over 80 invited talks.