symmetry

MDPI

Article

A Novel Video Steganography-Based Botnet Communication Model in Telegram SNS Messenger

Minkyung Kwak and Youngho Cho *

Department of Defense Science (Computer Engineering and Cyberwarfare Major), Graduate School of Defense Management, Korean National Defense University, Nonsan 33021, Korea; kmk85@kndu.ac.kr
* Correspondence: youngho@kndu.ac.kr

Abstract: In botnets, a bot master regularly sends command and control messages (C & C messages) to bots for various purposes, such as ordering its commands to bots and collecting critical data from bots. Although such C & C messages can be encrypted by cryptographic methods to hide them, existing botnet detection mechanisms could detect the existence of botnets by capturing suspicious network traffics between the bot master (or the C & C server) and numerous bots. Recently, steganography-based botnets (stego-botnets) have emerged to make C & C communication traffics look normal to botnet detection systems. In stego-botnets, every C & C message is embedded in a multimedia file, such as an image file by using steganography techniques and shared in Social Network Service (SNS) websites (such as Facebook) or online messengers (such as WeChat or KakaoTalk). Consequently, traditional botnet detection systems without steganography detection methods cannot detect them. Meanwhile, according to our survey, we observed that existing studies on the steganography botnet are limited to use only image steganography techniques, although the video steganography method has some obvious advantages over the image steganography method. By this motivation, in this paper, we study a video steganography-based botnet in Social Network Service (SNS) platforms. We first propose a video steganography botnet model based on SNS messengers. In addition, we design a new payload approach-based video steganography method (DECM: Divide-Embed-Component Method) that can embed much more secret data than existing tools by using two open tools VirtualDub and Stegano. We show that our proposed model can be implemented in the Telegram SNS messenger and conduct extensive experiments by comparing our proposed model with DECM with an existing image steganography-based botnet in terms of C & C communication efficiency and undetectability.

Keywords: botnet; steganography botnet; telegram; video steganography; SNS security

check for updates

Citation: Kwak, M.; Cho, Y. A Novel Video Steganography-Based Botnet Communication Model in Telegram SNS Messenger. *Symmetry* **2021**, *13*, 84. https://doi.org/sym13010084

Received: 10 December 2020
Accepted: 4 January 2021
Published: 6 January 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Cyberattacks evolve to avoid or nullify detection methods of existing security systems. Recent botnets also evolve to hide their command and control messages (C & C messages) to avoid being detected by existing botnet detection systems [1,2]. Recently, a novel type of botnet using steganography techniques has emerged to hide the existence of C & C communication itself, which is the so-called *steganography-based botnet* or *stego-botnet* [3,4]. In particular, when stego-botnets are constructed in Social Network Service (SNS) platforms, it becomes much more difficult to detect the stego-botnets since every botnet C & C communication message is hidden into a multimedia file (e.g., image file), which look normal to users in SNSs.

Meanwhile, most existing studies on stego-botnets are limited to using image steganography techniques because of the simplicity of adopting those techniques and the popularity of sharing image files in the SNS [3–5]. However, in addition to an image file, since there are various cover mediums, such as a video file, an audio file, and document files, including HTML, various steganography techniques depending on the types of cover mediums can

be used in stego-botnets [6–9]. Especially, a video file is a very attractive cover medium because it is not only actively shared in SNSs (i.e., not suspicious to users), but also has a big volume of payload that can be considered for data hiding compared to other types of cover medium. Thus, there are clear advantages of using video steganography methods over image steganography methods in terms of embedding capacity and anti-steganalysis [10,11]. By this motivation, we in this paper study video steganography botnets in SNSs.

Our contributions in this paper can be summarized as follows.

- We proposed the first video steganography-based botnet model that can be constructed in an SNS messenger, and implemented its core part at the real Telegram SNS messenger.
- We devised a new video steganography method (DECM: Divide-Embed-Combine Method) based on two open tools (VirtualDub [12] and Stegano [13]) that can embed secret data into payloads of a cover video file much more than existing video steganography tools can.
- We validated that our proposed model and method are more efficient than an image steganography-based botnet model, in terms of the number of cover medium files used, which is necessary to embed the same amount of secret data to be embedded. Thus, the lower the number of cover medium files, the higher the undetectability of a C & C message in a botnet.

By reporting our study to the academia in the security field, we hope that this study can provide useful information about the advanced new botnet C & C model, which may appear in real cyberattacks or cybercrimes, raise an alarm to security engineers and researchers, and, thus, attract them to research effective defense mechanisms and techniques against the botnet model.

The rest of our paper is organized as follows. In Section 2, we overview traditional botnets and steganography-based botnets and introduce existing studies related to them. In Section 3, we propose the first video steganography-based botnet model in an SNS messenger. In Section 4, we devise a new video steganography method (DECM: Divide-Embed-Combine Method). In Section 5, we implement the core part of our model at the Telegram Messenger, and conduct extensive comparative experiments to show the performance of our model in Section 5. We conclude in Section 6.

2. Background and Related Works

2.1. Traditional Botnet

A botnet is a network of bots that are maliciously infected computing devices with network functions and under the control of a bot master. In general, the traditional botnet consists of three main components: Bot master, C & C server, and bots (see Figure 1) [1,14]. The bot master is a cyber-attacker that controls the botnet, and the C & C server is a command and control server that receives commands from the bot master, and delivers the commands to the bots or deliver information collected from the bots to the bot master; a bot master and C & C server can be combined. The bots conduct malicious activities, such as Distributed Denial of Service (DDoS) attacks according to the bot master's commands [15]. Therefore, the number of bots will affect the impact of the malicious attacks performed by the botnet, and social engineering techniques such as phishing with drive-by download are actively used to attract and recruit the bots [16].

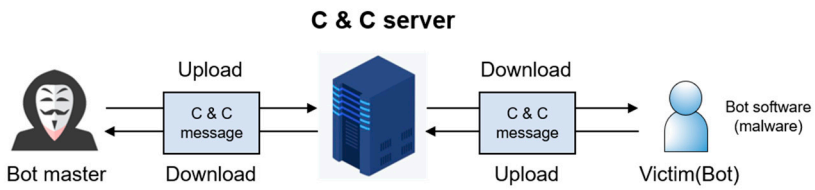


Figure 1. The general structure and major components of traditional botnets.

To operate well the botnet, the bot master and bots must exchange C & C messages via the C & C server. Although such C & C message packets can be encrypted [17,18] or piggy-backed over some network protocols [19] to hide them against botnet detectors, recent advanced botnet detection systems can capture the existence of botnets by using sophisticated network traffic analysis methods [20–22].

2.2. Steganography-Based Botnet (Stego-Botnet)

As the popularity of SNS grows, many studies on constructing botnets in SNS platforms have been introduced. Wu et al. [23] proposed ServerLess botnet (SLbot) that uses an SNS platform for the C & C server and three types of C & C channels, such as the addressing channel, the command channel, and the upload channel. In addition, Faghani and Nguyen [24] proposed a cellular botnet, which is called SoCellBot that recruits bots from SNS and uses SNS messengers for C & C channel between a bot master and a bot.

Recently, a novel type of botnet using steganography techniques (steganography-based botnet or stego-botnet) has emerged to avoid botnet detection methods used in traditional botnets [3,4]. The stego-botnets can avoid the existing detection methods by making botnet C & C messages look normal to them by using steganography techniques. Specifically, they hide all C & C messages into plain multimedia files, such as image or text files. Since they are usually constructed in an SNS homepage or an SNS messenger, existing botnet detection methods just observe that multimedia files are shared in the SNS, but cannot detect the existence of C & C messages embedded in those multimedia files.

There are a couple of studies on the stego-botnet that applies image steganography techniques to hide C & C communications via popular SNS services. Nagaraja et al. [3] proposed Stegobot, which is the first stego-botnet, based on image steganography and constructed on Facebook. Stegobot implements a distributed C & C communication channel through which compromised bots share digital images with secret messages in Facebook. In addition, Stegobot uses two types of C & C messages: (1) a bot-command broadcasts the bot master's commands to the bots, and (2) a bot cargo message delivers critical information of the bots to the bot master, according to bot-commands. Stegobot can transmit a C & C message whose size is lower than 40,280 bits (≈5 KBytes) per image and, thus, it is difficult to transmit a relatively large size of C & C messages. For the first stego-botnet using an SNS messenger platform, Jeon and Cho [4] introduced an image stego-botnet in the KakaoTalk SNS messenger. They implemented a part of image stego-botnet in the KakaoTalk messenger and demonstrated a C & C message can be transferred secretly from a bot master to a bot via a KakaoTalk chatroom. They also proposed a method that can increase the delivery rate of C & C messages in case that some participants do not read and download stego-images at the chatroom. Park and Cho [5] proposed an automated inspection system that detects steganography image files shared in SNS chatrooms. The proposed system semi-automatically collects and inspects all image files shared in an SNS chatroom based on multiple open image steganography tools.

Meanwhile, existing studies mainly focused on image-based stego-botnets in SNSs. However, to the best of our knowledge, there are no existing works on studying video steganography-based botnets. Video files are popularly shared among people in SNS services and compared to image files, they have advantages such that they have larger payloads and, thus, can contain larger secret data, and they are also known to be more resistant to anti-steganalysis [25,26].

Consequently, by this motivation, we in this study propose a novel botnet model based on video steganography techniques, verify whether the botnet communication can be implemented on a real SNS (the Telegram Messenger), and validate the advantage of using the video stego-botnet in SNS platforms in terms of C & C communication efficiency and undetectability.

3. Proposed Model: Video Steganography-Based Botnet Model in an SNS Messenger

3.1. Model Description

We propose a novel video steganography botnet model that can be constructed and implemented in an SNS messenger. As shown in Figure 2, this model has four main components: (1) bot master, (2) bots (victims), (3) SNS messenger, and (4) stego-video file. In this model, the SNS messenger plays a role as the C & C server of the traditional botnet model. However, it does not actively communicate with bots, but it simply acts as a public C & C message sharing platform from which bots freely download and upload C & C messages hidden in multimedia files. By this manner, this model can hide exchanging C & C messages against traditional botnet detection systems.

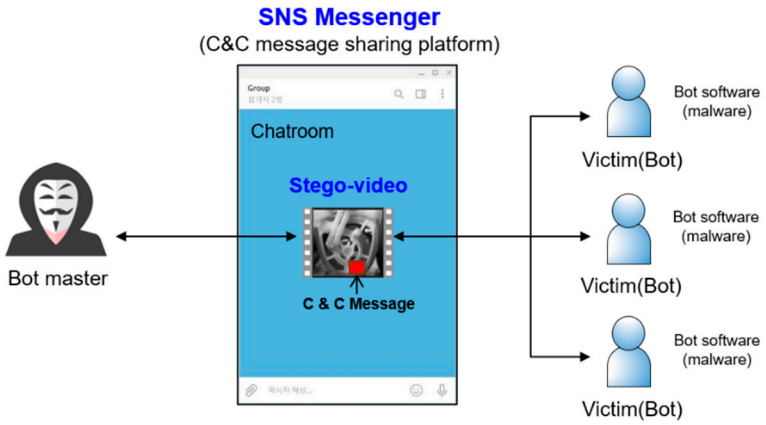


Figure 2. Video steganography-based botnet model in an in Social Network Service (SNS) messenger.

This model works as the following steps. We assume that bots are already compromised during the botnet construction stage and bot software is installed in their devices [27,28].

(1) The bot master prepares a video file (e.g., MPEG (Moving Picture Experts Group) video clip) as a cover medium that can contain a C & C message and then embeds the C & C message into the video file by using some video steganography tools (e.g., OpenPuff) or video steganography algorithms. In this paper, we call a video file with the hidden message a stego-video.

(2) The bot master creates a public chatroom or logins in an existing public chatroom with many participants (victims) in an SNS messenger and then it uploads the stego-video to the chatroom. By this manner, the stego-video is shared with all participants in the chatroom.

(3) The stego-video file is downloaded to participants' devices (e.g., smartphones or laptops), and bots (bot software) work according to the C & C message hidden in the stego-video.

3.2. Suitability of Telegram Messenger for C & C Message Sharing Platform

To be suitable for the C & C message sharing platform in our model, it is necessary for an SNS to have the following two properties: (1) every stego-video must be delivered to bots, and (2) embedded message hidden into the stego-video can be extracted correctly when they are delivered to bots. According to our examination on five popular global SNS messengers (see Table 1), we found that Telegram Messenger satisfies the above two properties and, thus, well fits the C & C messenger sharing platform in our model. We will explain more details in Section 5.1. In addition to the above basic properties, the Telegram Messenger has a couple of desirable aspects that make it suitable for our model as below.

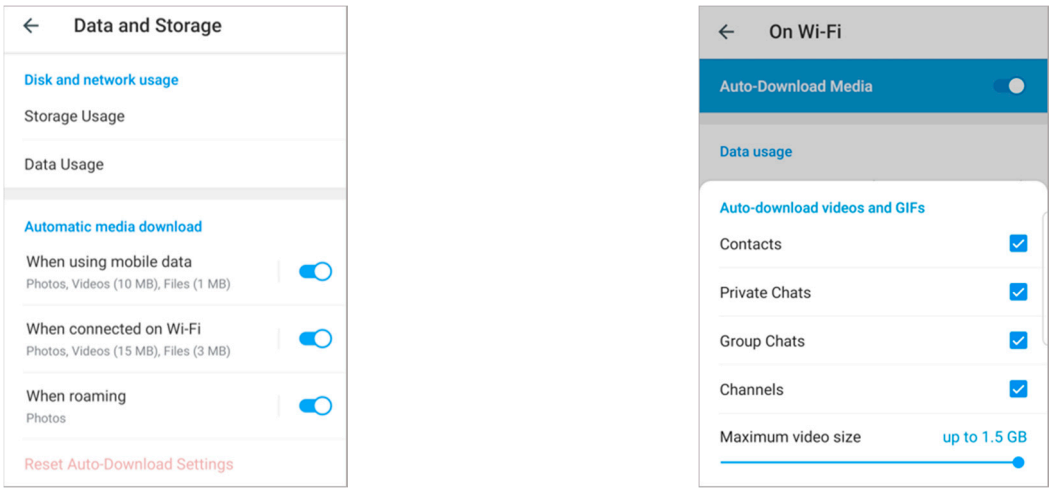
Table 1. Suitability check results for well-known Social Network Service (SNS) messengers.

	Telegram	**WhatsApp**	**Facebook**	**WeChat**	**KakaoTalk**
Version (desktop/android)	2.1/6.1	2.2027.10/2.20.194.16	Web/ 272.0.0.14.119	2.9.5.41/ 7.0.15	3.1.4.2500/ 8.9.3
Original video sharing function	O	X	X	X	X
Automatic download function	O	O	X	O	X
Maximum participants	200,000	256	15	500	1500
Maximum video size	1.5 GB	64 MB	25 MB	100 MB	300 MB

First, Telegram is one of the most popular global SNS messengers and it has about 400 million active monthly users worldwide. Since the higher number of bots (victims), the more powerful attack the botnet can conduct, it is advantageous for an SNS messenger to have numerous users. In addition, Telegram supports various types of chatroom communication options such as 1: 1, 1: N (channel), and N: N (group). For example, the channel and group communication options allow a participant (or bot master) to share files with many users who participate in a chatroom. For example, in the channel type, only the chatroom creator has the right to send messages or files (one-way communication) and the number of participants (or subscribers) that can participate in the channel is unlimited theoretically. In the case of the group chatroom, all participants can share messages and files (bi-directional communication) and the maximum number of participants is 200,000, which is a much larger number of participants compared to other SNS messengers as you can see in Table 1.

Second, Telegram users can share a very large size of video file up to 1.5 GB. According to our investigation, most SNS messengers strongly limit the size of a video file that can be shared at a chatroom due to various operational reasons. For example, as shown in Table 1, WhatsApp, which has the largest number of users in the world, can share a video file whose size is up to only 64 MB at a time and Facebook supports only 25 MB file at most. KakaoTalk supports a slightly larger size of video file than WhatsApp, Facebook, and WeChat, but it is limited to 300 MB. On the other hand, Telegram can share a video file up to 1.5 GB but this generous setting for video sharing has been exploited as a means of cybercrime such as a sexual exploitation case of children in Korea. Moreover, a bot master may embed a huge size of a secret message into such a large cover video file and then shares it in a Telegram chatroom.

Third, Telegram supports auto-download function of multimedia files including video files. At the Telegram android app's default setting, a video file whose size is less than 10 MB is automatically downloaded to a user's device when the user uses mobile data connection, and when Wi-Fi connection is available, a video file less than 15 MB is automatically downloaded (see Figure 3a). However, as shown in Figure 3b, if a user sets the value of maximum video size to its maximum (=1.5 GB), all video files less than 1.5 GB can be downloaded automatically to the user's device that uses Wi-Fi connection. Thus, a user (a bot or a victim) may automatically save a stego-video file to its device by simply viewing a video shared at a Telegram chatroom even without clicking the file. Even when the value of maximum video size is not set to 1.5 GB, it is still possible that a bot software installed in the victim's mobile phone may be able to change the value to 1.5 GB through unauthorized access to the Telegram App or privilege escalation.



(**a**) (**b**)

Figure 3. Telegram settings related to automatic video download (**a**) Telegram's auto-download option; (**b**) Telegram's maximum video size option.

3.3. Attack Scenarios

We now explain two attack scenarios in our proposed botnet model to help understand how this model can be used in launching cyberattacks.

First, the bot master can launch DoS attacks to some target server (or service). The bot master creates a stego-video file that contains DoS attack operational information, such as target server (or service), attack date and time, attack period and methods, and so on. Then, the bot master uploads it to a Telegram chatroom and it is automatically downloaded to all bots in that chatroom. After that, as shown in Figure 4, a group of victims starts launching DoS attacks simultaneously to the target server according to the attack operation, which is extracted from the stego-video file. However, since the stego-video file looks innocuous and normal, existing botnet detection systems cannot defend against such attacks.

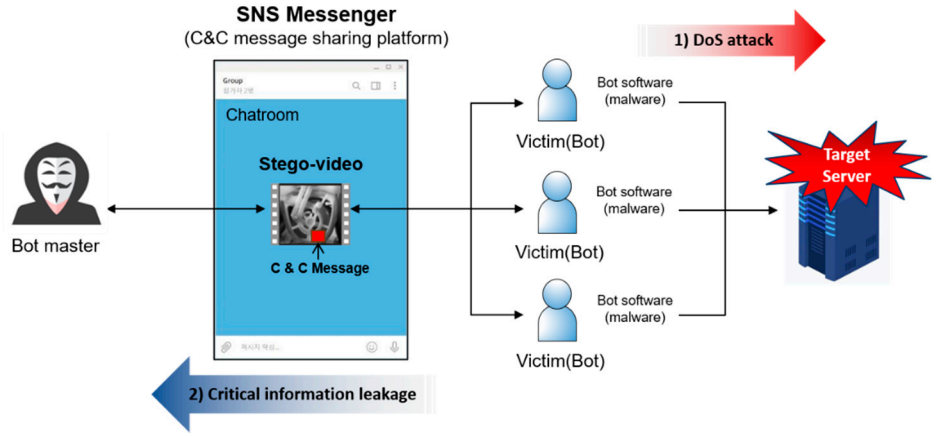


Figure 4. Attack scenario.

Second, the bot master can collect critical data from victims. The bot master may want to collect private data (such as passwords and SSN (Social Security Number)) from victims to conduct crimes to victims later or selling such data to criminal markets. The bot master creates a stego-video file that contains data collection methods, such as data of interest, collection period, and so on. As shown in Figure 4, after victims receive the stego-video

file, bot software extracts C & C messages from stego-video. Moreover, according to the C & C messages, they will collect and report data to the bot master without their notices.

4. Design of a New Video Steganography Method (DECM: Divide-Embed-Combine Method)

4.1. Motivation

In general, a video steganography method (or tool) is implemented in either the metadata approach or the payload approach. In the metadata approach, secret data is hidden in the header or footer part of a cover medium, such as an MP4 video file. For example, a secret message is hidden by modifying the metadata in the file header or embedded at the end of a file (EoF method). On the other hand, in the payload approach, secret data are hidden in the payload part of a cover medium. For example, the Lease Significant Bit (LSB) substitution method is one of the representative payload approach-based methods, in which each LSB is replaced with 0 or 1, according to the binary sequence of a hidden message to be embedded.

Since a video file consists of many connected image frames, video files in general have a much larger size of payloads compared to image files. Based on this fact, we infer that a video file has a much larger space in which a payload approach-based steganography method can embed a secret message than image files. In other words, a video file has a higher embedding capacity than an image file. By this inference, we claim that a video steganography tool will have much higher embedding efficiency than an image steganography tool has.

Meanwhile, to the best of our knowledge, most video steganography tools (OpenPuff, TcSteg, StegoStick, and so on) are based on the metadata approach. We could find only one payload approach-based video steganography, which is the MSU StegoVideo [29]. However, unlike our above inference, according to our test, MSU StegoVideo's average Peak Signal to Noise Ratio (PSNR) value of each frame dropped to 20 dB when the cover file size was 50 MB, and an embedding error occurred when the cover file size was 100 MB or more. This means that it is not feasible for using it in our video steganography botnet model.

By this motivation, we devise a payload approach-based new video steganography method (DECM: Divide-Embed-Combine Method) by using two existing open tools (VirtualDub and Stegano). As the name of our method indicates, the basic concept of devising our method is that we divide a cover video into image frames, embed secret data into the divided image frames, and combine all stego-image frames into a stego-video file (see Figure 5). We will explain in detail the working steps of our method in Section 4.2.

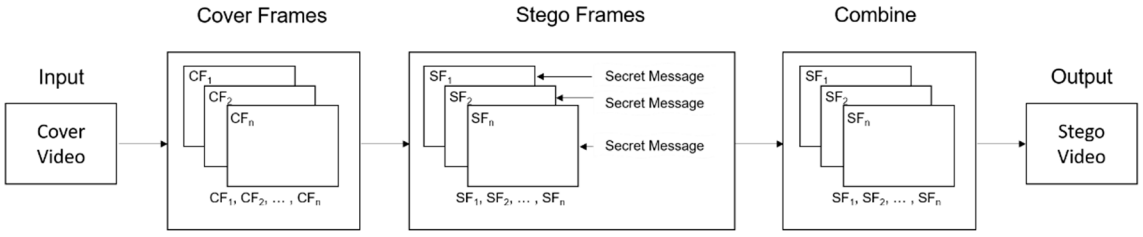


Figure 5. The basic concept of Divide-Embed-Combine Method (DECM).

4.2. Working Steps and Design of Proposed Method

The working steps of the proposed method are as follows (see Figure 6 and Algorithm 1).

- Step 1: Read a cover video file (e.g., AVI file);
- Step 2: Extract cover image frames from the cover video file by using VirtualDub;
- Step 3: Create stego-image frames by hiding secret data into the extracted images frame by using Stegano;

- Step 4: Combine all stego-image frames by using VirtualDub;
- Step 5: Produce the stego-video file (e.g., AVI file).

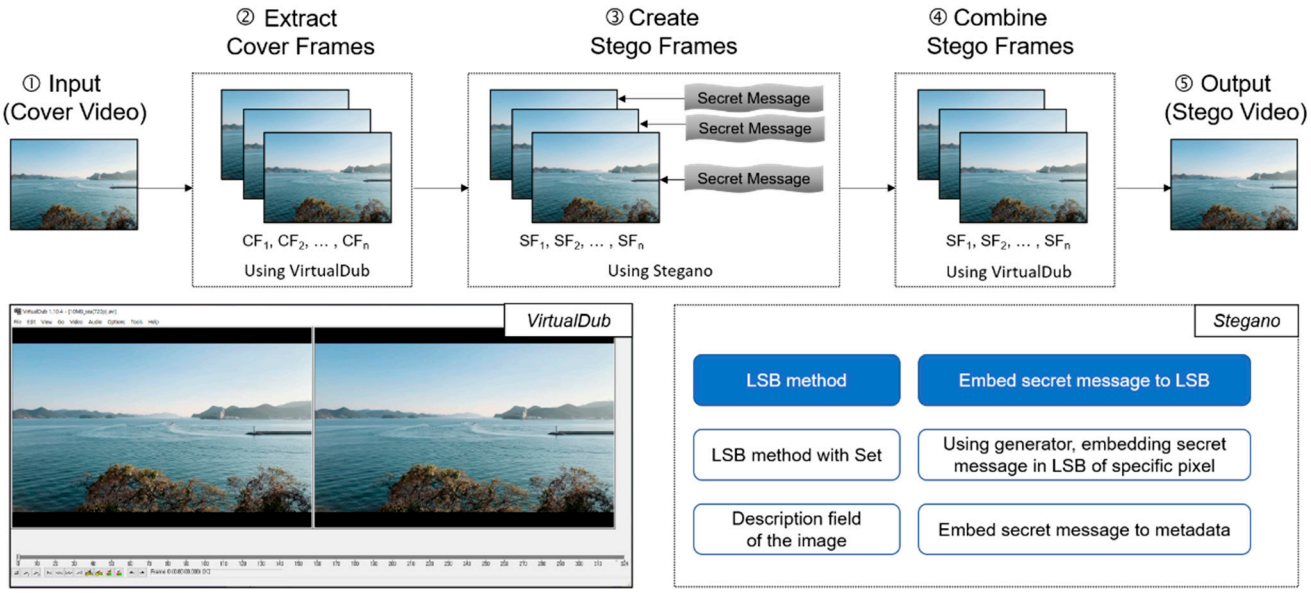


Figure 6. Working steps of Divide-Embed-Combine Method (DECM).

We implemented DECM by using a laptop (CPU: Core i5-8265U CPU, RAM: 8 GB, and OS: Windows 10). For image frame extraction (Step 2) and reassembly (Step 4), we used VirtualDub v1.0 program. In addition, to embed secret messages in extracted cover image frames, we implemented Python codes based on Stegano v0.9.8 library. In addition to Stegano, other open image steganography tools can be considered for Step 3 in our DECM.

Algorithm 1: Divide-Embed-Combine Method (DECM)

Input
Cover video file V_C
Secret message M_S
Output
Cover image frames $CF = \{CF_1, CF_2, \ldots, CF_N\}$
Partitioned secret messages $SM = \{SM_1, SM_2, \ldots, SM_N\}$
Stego-image frames $SF = \{SF_1, SF_2, \ldots, SF_N\}$
Stego-video file SV

begin
  read V_C
  $CF \leftarrow$ extract image frames from V_C by using VirtualDub
  partition M_S
  for each CF_i **in** CF:
    $SF_i \leftarrow$ embed SM_i into CF_i by using Stegano
  $V_S \leftarrow$ combine $SF_1, SF_2, \ldots, SF_N$ by using VirtualDub
end

5. Experiment Results

In this section, we conduct two kinds of experiments to show that (1) our proposed model can be implemented in a real SNS messenger (Telegram) in Experiment 1 and (2) our proposed model (video-stego botnet) has some advantages over the existing image-stego botnet model in terms of botnet C & C message communication efficiency in Experiment 2.

5.1. Experiment 1

5.1.1. Experimental Purpose and Methods

The goal of Experiment 1 is to validate that our proposed video steganography-based botnet model can be implemented in the Telegram SNS messenger on the real Internet environment. In this experiment, we conclude that our model is valid if a stego-video file, which is shared in a Telegram chatroom, can be downloaded to a chatroom participant and then a secret message hidden stego-video file can be extracted at the participant's device without any modification.

For our experimental environment setup, we used one laptop with Telegram Desktop App version 2.1 for the bot master ($N_{BOTMASTER}$) and one smartphone device with Telegram Android version 6.1 for the victim (N_{VICTIM} or N_{BOT}). We used Telegram's default settings.

For cover video files, we used five different video files downloaded from the web. To create stego-video files, we embedded a secret message "*attack*" into them by using four well-known, free steganography tools (OpenPuff [30], TcSteg [31], StegoStick [32], and MSU StegoVideo [29]), as well as our devised hiding method DECM. Since each tool supports different video formats, we created 12 stego-video files with five different video formats (FLV, MP4, 3GP, VOB, and AVI).

We conducted Experiment 1 as follows (see Figure 7).

(1) $N_{BOTMASTER}$ (laptop) creates a chatroom in the Telegram Messenger.
(2) $N_{BOTMASTER}$ (laptop) requests N_{VICTIM} (smartphone) to participate in the chatroom.
(3) N_{VICTIM} (smartphone) joins the chatroom.
(4) $N_{BOTMASTER}$ (laptop) uploads all stego-video files (12 Vs) one by one into the Telegram chatroom.
(5) N_{VICTIM} (smartphone) reads all shared video files. In this step, when N_{VICTIM} simply reads them, all stego-video files shared at the chatroom are automatically downloaded to the local storage of N_{VICTIM}'s device (smartphone). We checked this by using digital forensic methods and will explain in detail later.
(6) We locate downloaded stego-video files at N_{VICTIM}'s device and examine, try to extract a hidden message from the stego-video files.
(7) We check if extracted messages from the stego-video files match the original secret message "attack". If matches, we conclude our proposed model can be implemented in the Telegram Messenger.

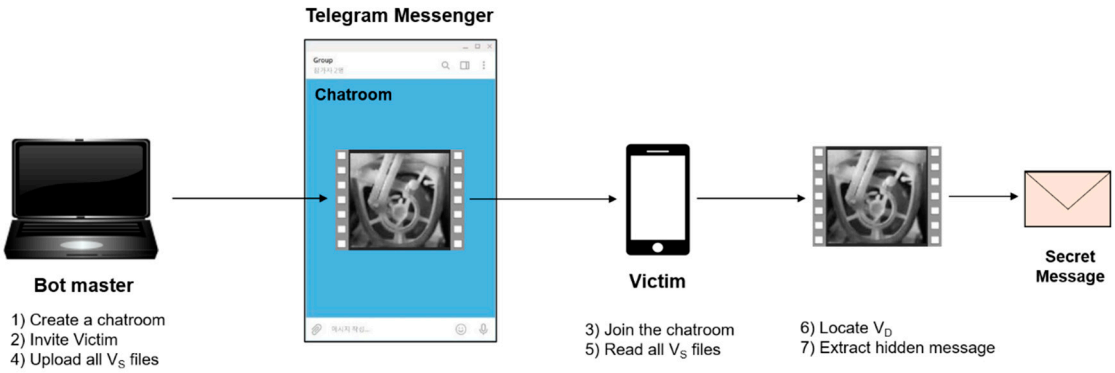


Figure 7. Experimental procedures of Experiment 1.

5.1.2. Experimental Results

We now explain our experimental results and findings as follows.

First, all stego-files (12 V_S) were successfully delivered from the bot master ($N_{BOTMASTER}$) to the victim (N_{VICTIM}) without any failures and modifications (See Table 2). This must be satisfied for our proposed model to be valid because if a stego-video file is not delivered to the bot (the victim), that means the botnet C & C message embedded into the stego-video

file cannot be used at the bot node. As shown in Figure 8, by using HashMyFiles [33], we examined that the hash values of sending stego-video files (V_S) and received stego-video files (V_D) were identical.

Table 2. Experimental results of experiment 1 (support/result).

Tools	FLV	MP4	3GP	VOB	AVI
OpenPuff (v4.0.1)	O/ Success	O/ Success	O/ Success	O/ Success	X/-
TcSteg (v3.0)	X/-	O/ Success	X/-	X/-	X/-
StegoStick (v1.0)	O/ Success	O/ Success	O/ Success	O/ Success	O/ Success
MSU StegoVideo (v1.0)	X/-	X/-	X/-	X/-	O/ Success
DECM	X/-	X/-	X/-	X/-	O/ Success

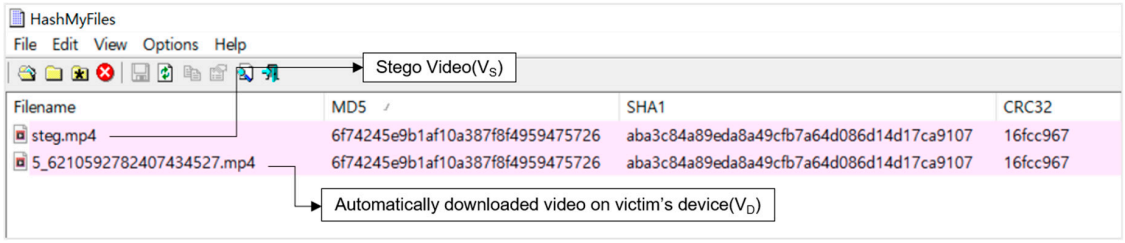


Figure 8. Hash values of an uploaded file (V_S) and a downloaded file (V_D).

Second, we could locate all stego-video files at the victim's smartphone storage (directory path: Telegram\Telegram Video or Telegram\Telegram Document) by using digital forensic techniques. This is well supported by the existing work [34]. All stego-video files shared in the Telegram chatroom were automatically downloaded to the victim's smartphone, although the victim did not save those files but simply read them on the Telegram chatroom screen.

Third, the hidden messages from all delivered stego-video files (12 V_D) were successfully extracted and match the original hidden secret message "*attacks*." For example, Figure 9 shows that the hidden message "attack" was correctly extracted from the downloaded stego-video file that corresponds to the stego-video file (sea.mp4) generated by OpenPuff.

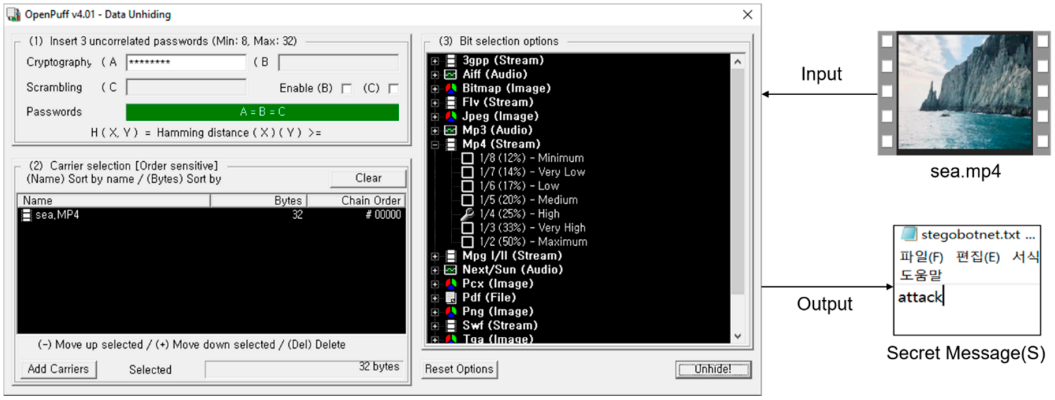


Figure 9. Successful extraction of a hidden message "attack" from the downloaded video file (sea.mp4) by using OpenPuff.

Therefore, based on our experiment results, we validated that the implementation and construction of our proposed model can be possible in the Telegram SNS messenger and it works properly.

5.2. Experiment 2

5.2.1. Experimental Purpose and Methods

In experiment 2, we conduct a comparative analysis to show a video stego-botnet is more efficient and undetectable than an image stego-botnet in terms of botnet C & C communication. For comparison, we consider the following: given two steganography methods (method i and method j) and a secret message m, method i is more efficient and undetectable than method j when method i requires less stego medium files to send m to bots than method j. This is reasonable in that the larger the number of stego cover mediums to send the same amount of secret data, the less efficient in terms of communication and the more suspicious to detection systems.

Based on the above claim, we conduct experiment 2 as follows.

First, we define a metric *NCMF(m)* as the number of cover medium files to send a secret message m and measure it for comparative analysis. *NCMF(m)* can be obtained by using two sub-metrics Maximum Embedding Capacity (*MEC*) and Peak Signal to Noise Ratio (*PSNR*) for both our method and existing method. *MEC(f)* is the maximum embedding capacity that a cover medium file f can have, and given two image files A (cover image) and B (stego image), *PSNR*, which is a well-known metric in an image processing area, can be obtained as

$$PSNR = 10 \times log_{10}\left(\frac{MAX_A^2}{MSE}\right)(dB) \quad (1)$$

where $MSE() = \frac{\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c}[A(i,j,k)-B(i,j,k)]^2}{a \times b \times c}$, a and b represent the resolution of the image (frame), c represents the RGB color component, and MAX_A is the maximum pixel value of the image (frame) A [35]. In general, when *PSNR* is higher than 30 dB, it is known that two images are distinguishable with human eyes and thus it is very difficult for human eyes to detect the stego file [36]. Thus, by using *MEC* and *PSNR*, we will measure *NCMF(m)* of both our method and image-based method fairly and we will explain in more detail below.

Next, we conducted experiment 2 in the following steps (step 1–step 3).

In step 1, we collected sample cover images and videos from the real chatroom for comparison. For a fair comparison, we collected 500 image and 500 video sample files from five real Telegram chatrooms with more than 1000 participants.

In step 2, we measured the average *MEC* for the collected samples from step 1. For a secret message for embedding, we generated a text file with random character sequences that consist of "a" ~ "z" and "." as necessary. In addition, for data embedding tools, we used OpenStego [37] and Steg [38] for image files and MSU StegoVideo, and our devised DECM method for video files. To obtain *MEC*, we embedded data into a cover file as much as we can while satisfying PNSR $\geq$ 30 dB to avoid embedding data into cover files excessively.

In step 3, we calculated and compared the values of *NCMF* of both methods. According to various sizes of secrete messages ranging from 1 MB to 3 GB, we checked how many cover images and videos are required based on *MEC*.

5.2.2. Experimental Results

We report our experimental result and analysis according to the experimental steps.

First, Table 3 shows the measured average, minimum, and maximum file size of sample images and videos collected in a real Telegram chatroom in step 1. Among those statistics, we will use the average file size of sample files for comparison in the next steps. The average file size of 100 image samples is 0.1 MB and the average file size of 100 video samples is 217 MB. As we expected, the average file size of video samples is much larger than that of image samples.

Table 3. Average, maximum, and minimum size of sample images and videos collected in step 1.

Sample Type	AVG	MIN	MAX	SD
Image	0.1 MB	0.01 MB	0.3 MB	0.06 MB
Video	217 MB	0.1 MB	1199 MB	243.1 MB

Second, Table 4 shows calculated *MECs* when image and video steganography methods (OpenStego, Steg, MSU StegoVideo, and our DECM) are used. For an image sample, two image steganography tools (OpenStego and Steg) could embed secret data into a sample image file similarly. Specifically, OpenStego could embed 0.057 MB and Steg could embed 0.05 MB into a sample image file of 0.1 MB by satisfying various conditions such as PSNR $\geq$ 30 db we mentioned above. On the other hand, for a video sample, we used MSU StegoVideo and our DECM. During our experiment, MSU StegoVideo could not embed data into the sample video file by satisfying PSNR $\geq$ 30 dB, and it generated an error for unknown reasons; we tried to embed data into other sample videos whose sizes are 50 MB or 100 MB but failed in embedding data while satisfying PSNR $\geq$ 30 dB. Meanwhile, our DECM could embed 599 MB into the sample video successfully while keeping PSNR $\geq$ 30 dB.

Table 4. Maximum Embedding Capacity (MEC) of each method for sample image and video files.

Image		Video	
OpenStego	**Steg**	**MSU StegoVideo**	**DECM**
0.057 MB	0.05 MB	NA	599 MB

Last, Table 5 and Figure 10 show measured *NCMF* according to image and video steganography methods, and for all sizes of secret data ranging from 1 MB to 3 GB, our devised DECM method requires much less *NCMF* than two image-based methods. For example, when a bot master wants to deliver 1 MB of secret data to bots via an SNS chatroom, OpenStego-based image-botnet needs 18 image cover files and Steg-based image-botnet needs 20 image cover files. Meanwhile, our DECM-based video-botnet needs only one video cover file. In addition, when a bot master wants to deliver 3 GB of secret data to bots, OpenStego-based image-botnet needs 53,895 image cover files and Steg-based image-botnet needs 61,920 image cover files. Meanwhile, our DECM-based video-botnet needs only six video cover files. Thus, since *NCMF* indicates the number of cover medium files required between a bot master and bots through C & C communication, the smaller *NCMF* is the more botnet communication efficient and undetectable. Therefore, based on our experiment result, we validated that our proposed DECM-based video-botnet is more efficient in terms of botnet C & C communication and less suspicious to detection systems than image-based botnet.

Table 5. Number of Cover Medium Files (NCMF) measured for image and video steganography methods used in experiments.

Size of Secret Data	Image (OpenStego)	Image (Steg)	Video (DECM)
1 MB	18	20	1
5 MB	88	100	1
10 MB	176	200	1
20 MB	351	400	1
30 MB	527	600	1
50 MB	878	1000	1
100 MB	1755	2000	1
200 MB	3509	4000	1
300 MB	5264	6000	1
500 MB	8772	10,000	1
1024 MB	17,965	20,480	2
2048 MB	35,930	40,960	4
3072 MB	53,895	61,920	6

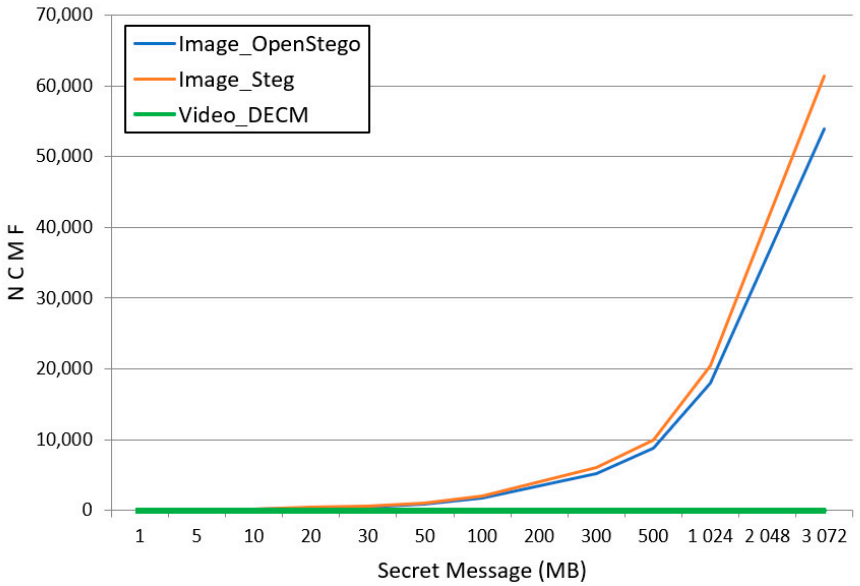


Figure 10. NCMFs measured for image and video steganography methods used in experiments.

5.2.3. Discussion

In this section, we briefly discuss the advantages and limitations of our proposed model in this paper.

First, our model has some advantages over the existing image steganography-based botnet model in SNS messengers, in terms of data hiding capacity and anti-steganalysis (undetectability) [39,40]. This is mainly because of the differences between image steganography and video steganography techniques. Specifically, our experiments showed that the embedding capacity of video cover mediums are much larger than that of the image cover mediums and, thus, given the same size of a secret message, the less number of video cover mediums are required in the video-based botnet approach than the image-based botnet approach was used as the cover medium. In addition, it is more difficult to detect a stego-video file than a stego-image. Thus, while one single image file needs to be examined to detect a stego-image file, it is necessary to use advanced steganalysis methods to detect a stego video file when a hidden message is scattered into multiple image frames of a video cover file randomly or in obfuscated manners [41–44].

Meanwhile, our model has a limitation such that our proposed model cannot be implemented in every SNS messenger. There exist a couple of SNS messengers that compress or process video files shared in a chatroom for some purpose of operational

efficiency (e.g., to save server storage or to improve the latency), which results in a critical data loss of the embedded secret message or even in the failure of extracting the embedded secret message itself. For example, the KakaoTalk messenger (ver. 9.1.6) does not support the original video sharing at the chatroom [45].

6. Conclusions and Future Works

In this paper, we first proposed a video steganography botnet model based on SNS messengers. Next, we designed a new payload approach-based video steganography method (DECM: Divide-Embed-Component Method) that can embed much more secret data than existing tools by using two open tools VirtualDub and Stegano. We showed that our proposed model can be implemented and work well in the Telegram SNS messenger and conducted extensive experiments that compare our prosed model with DECM with an existing image steganography-based botnet in terms of C & C communication efficiency and undetectability.

Our future research directions are as follows. First, we will extend our study by investigating other well-known, famous SNS messengers and report how our proposed botnet model can be serious in their operational environments. Next, we will design a more advanced type of steganography-based botnet that adaptively uses various types of cover mediums such as image, audio, video, or documents at the chatroom in SNS messengers. Last, we will devise effective defensive methods against steganography-based botnet models in the SNS messengers. This will include studying promising defensive measures to detect steganography botnet models in SNS messengers by devising new measures, such as NCMF, or adopting some effective measures used in traditional botnets, including encrypted botnets.

Author Contributions: Conceptualization, Y.C.; methodology, M.K. and Y.C.; software, M.K.; validation, M.K.; formal analysis, M.K. and Y.C.; investigation, M.K. and Y.C.; writing—original draft preparation, M.K. and Y.C.; writing—review and editing, Y.C.; visualization, M.K.; supervision, Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

Acknowledgments: A preliminary version of this paper was presented at the APIC-IST international conference, Seoul, Republic of Korea in 2020. The authors would like to thank the editor and reviewers for their valuable comments and constructive suggestions.

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

References

1. Zhuang, D.; Chang, J.M. Enhanced PeerHunter: Detecting Peer-to-Peer Botnets through Network-Flow Level Community Behavior Analysis. *IEEE Trans. Inf. Forensics Secur.* **2018**, *14*, 1485–1500. [CrossRef]
2. Gaonkar, S.; Dessai, N.; Costa, J.; Borkar, A.; Aswale, S.; Shetgaonkar, P. A survey on botnet detection techniques. In Proceedings of the 2020 International Conference on Emerging Trends in Information Technology and Engineering (IC-ETITE), Vellore, India, 24–25 February 2020.
3. Nagaraja, S.; Houmansadr, A.; Piyawongwisal, P.; Singh, V.; Agarwal, P.; Borisov, N. Stegobot: A covert social network botnet. In Proceedings of the 2011 International Workshop on Information Hiding, Berlin, Heidelberg, 18–20 May 2011.
4. Jeon, J.; Cho, Y. Construction and performance analysis of image steganography-based botnet in KakaoTalk openchat. *Computers* **2019**, *8*, 61. [CrossRef]
5. Park, J.; Cho, Y. Design and Implementation of Automated Steganography Image-Detection System for the KakaoTalk Instant Messenger. *Computer* **2020**, *9*, 103. [CrossRef]
6. Sun, Y.; Lu, Y.; Chen, J.; Zhang, W.; Yan, X. Meaningful secret image sharing scheme with high visual quality based on natural steganography. *Mathematics* **2020**, *8*, 1452. [CrossRef]
7. Alhaddad, M.J.; Alkinani, M.H.; Atoum, M.S.; Alarood, A.A. Evolutionary detection accuracy of secret data in audio steganography for securing 5G-enabled internet of things. *Symmetry* **2020**, *12*, 2071. [CrossRef]

8. Niu, K.; Li, J.; Yang, X.; Zhang, S.; Wang, B. Hybrid adaptive video steganography scheme under game model. *IEEE Access* **2019**, *7*, 61523–61533. [CrossRef]
9. Yuk, S.; Cho, Y. A Time-based dynamic operation model for webpage steganography methods. *Electronics* **2020**, *9*, 2113. [CrossRef]
10. Sadek, M.M.; Khalifa, A.S.; Mostafa, M.G.M. Video steganography: A comprehensive review. *Multimed. Tools Appl.* **2014**, *74*, 7063–7094. [CrossRef]
11. Xue, Y.; Zhou, J.; Zeng, H.; Zhong, P.; Wen, J. An adaptive steganographic scheme for H.264/AVC video with distortion optimization. *Signal. Process. Image Commun.* **2019**, *76*, 22–30. [CrossRef]
12. VirtualDub (ver. 1.10.4). Available online: https://sourceforge.net/projects/virtualdub/postdownload (accessed on 9 December 2020).
13. Stegano (ver. 0.9.8). Available online: https://pypi.org/project/stegano (accessed on 9 December 2020).
14. Liu, J.; Xiao, Y.; Ghaboosi, K.; Deng, H.; Zhang, J. Botnet: Classification, attacks, detection, tracing, and preventive measures. *Eurasip J. Wirel. Commun. Netw.* **2009**, *1*, 692654. [CrossRef]
15. Wang, P.; Sparks, S.; Zou, C.C. An advanced hybrid peer-to-peer botnet. *IEEE Trans. Dependable Secur. Comput.* **2010**, *7*, 113–127. [CrossRef]
16. Sood, A.K.; Zeadally, S.; Enbody, R.J. An empirical study of HTTP-based financial botnets. *IEEE Trans. Dependable Secur. Comput.* **2016**, *13*, 236–251. [CrossRef]
17. Zhang, H.; Papadopoulos, C.; Massey, D. Detecting encrypted botnet traffic. In Proceedings of the 2013 IEEE INFOCOM, Turin, Italy, 14–19 April 2013.
18. Patsakis, C.; Casino, F.; Katos, V. Encrypted and covert DNS queries for botnets: Challenges and countermeasures. *Comput. Secur.* **2020**, *88*, 101614. [CrossRef]
19. Alenazi, A.; Traore, I.; Ganame, K.; Woungang, I. Holistic model for HTTP botnet detection based on DNS traffic analysis. In Proceedings of the 2017 International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments, Vancouver, BC, Canada, 25–27 October 2017; pp. 1–18.
20. Homayoun, S.; Ahmadzadeh, M.; Hashemi, S.; Dehghantanha, A.; Khayami, R. BoTShark: A deep learning approach for botnet traffic detection. *Cyber Threat Intell.* **2018**, *70*, 137–153.
21. Mousavi, S.H.; Khansari, M.; Rahmani, R. A fully scalable big data framework for botnet detection based on network traffic analysis. *Inf. Sci.* **2020**, *512*, 629–640. [CrossRef]
22. Gezer, A.; Warner, G.; Wilson, C.; Shrestha, P. A flow-based approach for Trickbot banking trojan detection. *Comput. Secur.* **2019**, *84*, 179–192. [CrossRef]
23. Wu, D.; Fang, B.; Yin, J.; Zhang, F.; Cui, X. SLBot: A serverless botnet based on service flux. In Proceedings of the 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC), Guangzhou, China, 18–21 June 2018.
24. Faghani, M.; Nguyen, U. Mobile botnets meet social networks: Design and analysis of a new type of botnet. *Int. J. Inf. Secur.* **2018**, *18*, 423–449. [CrossRef]
25. Rabie, T.; Baziyad, M. The Pixogram: Addressing high payload demands for video steganography. *IEEE Access* **2019**, *7*, 21948–21962. [CrossRef]
26. Liu, S.; Xu, D. A robust steganography method for HEVC based on secret sharing. *Cogn. Syst. Res.* **2020**, *59*, 207–220. [CrossRef]
27. Vormayr, G.; Zseby, T.; Fabini, J. Botnet communication patterns. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2768–2796. [CrossRef]
28. Fedynyshyn, G.; Chuah, M.; Tan, G. Detection and classification of different botnet C & C channels. In Proceedings of the International Conference on Autonomic and Trusted Computing, Berlin/Heidelberg, Germany, 2–4 September 2011; pp. 228–242.
29. MSU StegoVideo (ver. 1.0). Available online: http://compression.ru/video/stego_video/index_en.html (accessed on 9 December 2020).
30. OpenPuff (ver. 4.01). Available online: https://embeddedsw.net/OpenPuff_Steganography_Home.html (accessed on 9 December 2020).
31. TcSteg (ver. 3.0). Available online: https://keyj.emphy.de/real-steganography-with-truecrypt (accessed on 9 December 2020).
32. StegoStick (ver. 1.0). Available online: https://sourceforge.net/projects/stegostick (accessed on 9 December 2020).
33. HashMyFiles (ver. 2.36). Available online: https://www.nirsoft.net/utils/hash_my_files.html/ (accessed on 9 December 2020).
34. Anglano, C.; Canonico, M.; Guazzone, M. Forensic analysis of Telegram messenger on android smartphones. *Digit. Investig.* **2017**, *23*, 31–49. [CrossRef]
35. Liu, Y.; Liu, S.; Wang, Y.; Zhao, H.; Liu, S. Video steganography: A review. *Neurocomputing* **2019**, *335*, 238–250. [CrossRef]
36. Sadek, M.M.; Khalifa, A.S.; Mostafa, M.G.M. Robust video steganography algorithm using adaptive skin-tone detection. *Multimed. Tools Appl.* **2017**, *76*, 3065–3085. [CrossRef]
37. OpenStego (ver. 0.7.3). Available online: https://github.com/syvaidya/openstego/releases/tag/openstego-0.7.3 (accessed on 9 December 2020).
38. Steg (ver. 1.1.0.0). Available online: https://www.fabionet.org (accessed on 9 December 2020).
39. Mstafa, R.J.; Elleithy, K.M.; Abdelfattah, E. A robust and secure video steganography method in DWT-DCT domains based on multiple object tracking and ECC. *IEEE Access* **2017**, *5*, 5354–5365. [CrossRef]
40. Cao, M.; Tian, L.; Li, C. A secure video steganography based on the intra-prediction mode (IPM) for H264. *Sensors* **2020**, *20*, 5242. [CrossRef]

41. Mstafa, R.J.; Younis, Y.M.; Hussein, H.I.; Atto, M. A new video steganography scheme based on Shi-Tomasi corner detector. *IEEE Access* **2020**, *8*, 161825–161837. [CrossRef]
42. Yao, Y.; Yu, N. Motion vector modification distortion analysis-based payload allocation for video steganography. *J. Vis. Commun. Image Represent.* **2021**, *74*, 102986. [CrossRef]
43. Yadav, P.; Mishra, N.; Sharma, S. A secure video steganography with encryption based on LSB technique. In Proceedings of the 2013 IEEE International Conference on Computational Intelligence and Computing Research, Enathi, India, 26–28 December 2013.
44. Ramalingam, M.; Isa, N.A.M. A data-hiding technique using scene-change detection for video steganography. *Comput. Electr. Eng.* **2016**, *54*, 423–434. [CrossRef]
45. KakaoTalk. Available online: https://cs.kakao.com/helps?service=8&category=24&locale=ko&device=1013&articleId=1073189039 (accessed on 1 September 2020).