Weaknesses of Popular and Recent Covert Channel Detection Methods and a Remedy

Sebastian Zillien and Steffen Wendzel, *Member, IEEE*

***Abstract*—Network covert channels are applied for the secret exfiltration of confidential data, the stealthy operation of malware, and legitimate purposes, such as censorship circumvention. In recent decades, some major detection methods for network covert channels have been developed. In this article, we investigate two highly cited detection methods for covert timing channels, namely ϵ-similarity and compressibility score from Cabuk et al. (jointly cited by 949 articles and applied by several researchers). We additionally analyze two recent ML-based detection methods: *GAS* (2022) and *SnapCatch* (2021). While all these detection methods must be considered valuable for the analysis of typical covert timing channels, we show that these methods are not reliable when a covert channel's behavior is slightly modified. In particular, we demonstrate that when confronted with a simple covert channel that we call ϵ-κlibur, all detection methods can be circumvented or their performance can be significantly reduced although the covert channel still provides a high bitrate. In comparison to existing timing channels that circumvent these methods, ϵ-κlibur is much simpler and eliminates the need of altering previously recorded traffic. Moreover, we propose an enhanced ϵ-similarity that can detect the classical covert timing channel as well as ϵ-κlibur.**

***Index Terms*—Anomaly detection, covert channel, GAS, information hiding, network security, SnapCatch, steganography.**

I. Introduction

Covert channels are undesired and stealthy communication channels that aid multiple cybercriminal activities. For instance, botnets can use them to hide their command and control channels and spyware can employ them to secretly exfiltrate stolen information like credentials or database content [1], [2], [3]. Moreover, covert channels can be part of DDoS attacks [4]. Alternatively, covert channels can also be used for legitimate purposes. For example, Wustrow et al. show an approach that uses a covert channel to circumvent state-sponsored censorship by enabling an "end-to-middle proxy"[5].

One specific type of covert channel is based on the *inter-arrival time* (IAT, sometimes also *inter-packet delay*, IPD). The IAT is the elapsed time between two succeeding network packets and each IAT represents a secret symbol. For instance, an IAT of 100 ms might indicate a '0' bit while an IAT of 200 ms might indicate a '1' bit. Several improvements of the plain IAT channel have been proposed, cf. [6], [7], [8], [9].

For such IAT covert channels, multiple heuristics exist. Among these are two highly cited ones by Cabuk et al. that were published between 2004 and 2009 [10], [11], [12]. Research work has improved over these algorithms during the succeeding decade, leading to mostly ML-based detection methods, such as the recent *GAS* [13] (RNN-based) and *SnapCatch* (based on image processing and ML) [14] methods that reach almost perfect detection quality.

Often, publications that present new covert channels are accompanied by detection approaches and algorithms. In several cases, such detection approaches can perform well in test scenarios that were evaluated by the authors. However, it is usually not considered that there might be other possibilities to impair the effectiveness of these detection algorithms, i.e., without even decreasing a covert channel's bandwidth. We address this topic as follows:

1) We demonstrate that both, the ϵ-*similarity* and the *compressibility score*, provide unreliable results when confronted with a slight modification of the standard covert timing channel, which we call ϵ-κlibur. In comparison to previous attempts such as JitterBug, MB-CTC and TRCC, ϵ-κlibur is much simpler and eliminates the requirement of altering pre-defined (legitimate) network traffic or a complex traffic generation framework.
2) We show that ϵ-κlibur can moreover degrade the performance of two novel detection approaches *GAS* and *SnapCatch*.
3) We propose an enhanced ϵ-similarity to replace the original heuristic. Our enhanced ϵ-similarity has shown that it can detect the standard covert timing channel as well as ϵ-κlibur.

Please note that it is *not* our major goal to provide a covert channel that circumvents *all* known covert timing channel detection methods but to enhance the understanding and limitations of the ϵ-similarity and the compressibility score.

The remainder of our paper is structured as follows. Section II presents fundamentals and Section III covers related work. Our ϵ-κlibur covert channel is presented in Section IV. We evaluate ϵ-similarity and compressibility score against ϵ-κlibur in Section V. Afterwards, we suggest an enhanced detection heuristic and evaluate it against ϵ-κlibur and TRCC in Section VI, we

Manuscript received 5 April 2022; revised 25 November 2022; accepted 30 January 2023. Date of publication 1 February 2023; date of current version 13 November 2023. We gratefully acknowledge the funding of the research training group SIVERT by the German federal state of Rhineland-Palatinate. *(Corresponding author: Sebastian Zillien.)*

Sebastian Zillien is with the Center for Technology and Transfer (ZTT), Hochschule Worms, 67549 Worms, Germany (e-mail: szillien@hs-worms.de).

Steffen Wendzel is with the Center for Technology and Transfer (ZTT), Hochschule Worms, 67549 Worms, Germany, and also with the Faculty of Mathematics & Computer Science, FernUniversität in Hagen, 58097 Hagen, Germany (e-mail: wendzel@hs-worms.de).

Digital Object Identifier 10.1109/TDSC.2023.3241451

further evaluate ϵ-κlibur against the ML-based detectors GAS and SnapCatch in Section VII. Section VIII concludes.

II. Fundamentals

In this section, we first cover the general concept of the related covert timing channels (Section II.A), followed by an explanation of the analyzed detection methods (Section II.B).

A. Timing Covert Channels

The considered timing covert channel of Cabuk et al. [10] modulates the IATs between consecutive network packets of a connection by intentionally delaying packets. The channel is a form of the so-called *inter-packet times* (or: *inter-arrival times*) hiding pattern [15] in network steganography, or – more generally – the *element positioning* pattern in steganography [16], as packets are “positioned” in time.

To transmit data, the covert sender and covert receiver first agree on two or more IATs corresponding to two or more secret symbols. The covert sender then encodes the message into a list of symbols. Each of these symbols is transmitted by waiting for a corresponding time after a packet has been sent, and then sending out the next network packet to reach a certain IAT. This is repeated until all symbols have been transmitted. In the setup of Cabuk et al. the covert channel sends data every τ and 2τ units of time, e.g., every $5\ ms$ and $10\ ms$, to encode two secret bits. Thus, the average IAT is $3\tau/2$.

B. Covert Timing Channel Detection Methods

In the original two papers published at ACM CCS in 2004 [10], ACM TISSEC/TOPS in 2009 [12] and a related dissertation [11], Cabuk et al. introduced and evaluated the two detection heuristics for IAT-based covert channels that we selected for our investigation. These three publications received a widespread influence in the field, as shown by their derivatives (see Sections III.A and III.B) and their citations: according to Google Scholar, the CCS’04 paper received 619 citations, the TISSEC paper 184 citations, and the dissertation 146 citations, summing up to 949 citations as of late January 2023.

These detection methods try to find patterns or structure in the IATs of the network packets. Regular network traffic has mostly random IATs that are influenced by the network hardware, used software, protocols, topology, and load. Traffic containing a covert channel will show a clear structure due to the artificial delays. Fig. 1 shows a scatter plot of the IATs of two network recordings. We can see that the IATs for the regular traffic form a single larger cluster, while the covert channel traffic forms two distinct clusters.

1) ϵ-Similarity: The first detection method by Cabuk et al. that we investigated is the ϵ-similarity. The goal of this heuristic is to quantify the “similarity” of the IATs in a network recording. The idea behind this is that regular traffic will have random timings, which are not too similar to each other, while covert channel traffic will have groups of rather similar IATs. By comparing the plots of Fig. 1, these differences in structure

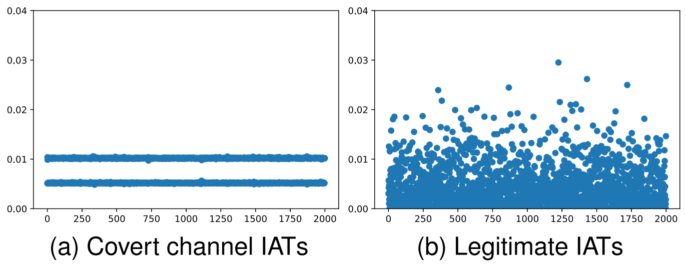


Fig. 1. Exemplary comparison of IATs.

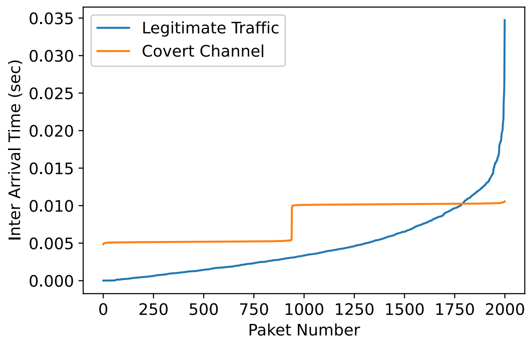


Fig. 2. Sorted IATs of the two-symbol covert channel and legitimate traffic.

are clearly visible. The ϵ-similarity is a numerical score that is calculated as follows:

1) Calculate all IATs for a given flow with 2,000 packets (called *window* size).
2) Sort the IATs (illustrated in Fig. 2).
3) Calculate the relative differences of two consecutive IATs:

$$\lambda_i = \frac{|t_{i+1} - t_i|}{t_i}$$

4) Calculate the percentage of λ values that are below the threshold ϵ, which is called the *similarity score*.

The similarity score is then used as a threshold to differentiate between legitimate and covert channel traffic.

2) Compressibility: The second detection approach introduced by Cabuk et al. is the compressibility score. The goal is again to quantify the structure of the IATs. This heuristic uses the help of a compression algorithm to approximate the Kolmogorov complexity [11], [17] of an IAT string-representation. The main functionality of the compression algorithm is to find patterns and structure in the data that can be exploited to efficiently compress the data. Highly structured data, like natural language texts or HTML files, can often be compressed with a high rate, while pseudo-random data, like encrypted data, can be compressed only slightly. The idea behind the compressibility heuristic is that the string representation of legitimate IATs will be more random and therefore less compressible, while covert channel traffic will have more structure and therefore be better compressible. By comparing the compression ratio of the string-representations of the IATs, we obtain a numerical measure of their “structure”. Consequently, legitimate traffic should have lower compressibility scores than covert channel traffic.

Cabuk et al. define the compressibility score as the compression rate that is calculated in the last step of the following algorithm:

1) Calculate all IATs for a given flow (window size: 2,000 packets).
2) Drop all values over 1.0 sec.
3) For each remaining IAT:
    1) Round to two significant digits.
    2) Convert the leading zeros after the decimal to a letter and concatenate it with the rounded value to a string s_i.
       As an example, 0.00346 sec would be transformed to B35 and 0.0346 sec to A35.
4) Concatenate all strings s_i to $S = s_1||s_2||\cdots||s_n$, with $||$ being the string concatenation.
5) Select a compressor $\Im$ (e.g., *gzip*) and calculate the compressed representation of S: $C = \Im(S)$.
6) Calculate the compressibility score $\kappa(S)$ as follows:

$$\kappa(S) = \frac{|S|}{|C|}$$

with $|x|$ representing the length of string x.

III. Related Work

In this section, we discuss related improvements of the standard version of the covert timing channel and how ϵ-κlibur is different from these approaches (Section III.A). Finally, we cover related detection heuristics (Section III.B).

A. Improvements to the Timing Covert Channel

Cabuk et al. further proposed ideas how to circumvent their detection methods by modifying the covert channel. But their approaches reduce the bandwidth of the covert channel by either mixing in legitimate traffic into covert transmissions en bloc or by significantly increasing the delays that are used for the encoding schema. Further, they proposed a more sophisticated timing channel called a *time-replay covert channel* (TRCC) in [11]. The TRCC functions similar to the basic IAT covert channel, but draws its delays from two (or more) sets of pre-recorded IATs of legitimate traffic. Each set corresponds to a secret symbol. However, Cabuk et al. did not perform an evaluation of the ϵ-similarity or the compressibility score with TRCC. Moreover, TRCC requires pre-recorded traffic, which ϵ-κlibur does not.

Related Timing Covert Channels: Groza et al. presented *JitterBug* [6], an optimized timing channel that takes legitimate Telnet traffic and adds random delays to the traffic so that it renders the traffic undetectable by multiple covert channel detection metrics. The covert channel was evaluated using keyboard input, which limits the scenario of JitterBug; the detectability using the ϵ-similarity and compressibility score were not evaluated.

Gianvecchio et al. introduced a *model-based covert timing channel* (MB-CTC) [7], which aims at evading detection by modeling and mimicking statistical properties of legitimate traffic. Therefore, the authors developed a framework that uses an appropriate distribution function in conjunction with a traffic library to build a covert channel. In comparison, we apply a much more simplified approach to create our channel and show that the two popular detection methods ϵ-similarity and compressibility score can be circumvented much easier than expected.

Zander et al. developed another sophisticated covert timing channel [18], which has shown low detectability in comparison to previous attempts. The detectability was evaluated using a Kolmogorov-Smirnov (KS) test and the C4.5 decision tree classifier.

Walls et al. proposed an improved covert timing channel called *Liquid* [8], which is based on JitterBug. Their goal was to further increase the covert channel's stealthiness when faced with entropy-based detection methods. The authors achieve this by splitting the channel in "transmitting" and "shaping" delays. The shaping delays carry no information but are used to manipulate the statistics of the transmission to more closely resemble the statistics of pre-recorded, legitimate traffic.

In 2015, Archibald and Ghosal presented a covert timing channel that is tailored to fit the behavior of Skype traffic [9]. To this end, and similarly to our approach, the authors applied multiple inter-packet delays. In comparison, our approach is generic (not tailored for a specific application, such as Skype) and targets different countermeasures, namely the ϵ-similarity or compressibility score.

It must be noted, that covert timing channels also appear in other forms and for other purposes. For instance, Lamshöft et al. recently utilized the timing of port knocking messages to influence `syslog` messages, so that they store secret information [19]. Moreover, the timing (as well as other metadata) of network traffic is actively manipulated for network flow watermarking [20], [21]. Further, there are timing-based covert channel patterns in addition to the inter-packet times or element positioning patterns [15], [16] analyzed in this paper, such as covert channels exploiting the timing of retransmissions [22], [23], covert channels influencing the throughput of a connection over time [24], and covert timing channels that drop selected network packets [25].

B. Improvements and Derivatives of the Detection Heuristics

Cabuk et al. proposed improvements of the compressibility score called *compressibility-walk* and *CosR-walk* to increase the detection performance when faced with mixed (covert and legitimate) traffic, which function as follows: The compressibility-walk uses a sliding window approach to evaluate a long flow of mixed covert channel traffic. For each window position, the compressibility score (κ) is calculated and then plotted. Windows which contain (only) covert channel traffic will be visible as peaks in this plot and could then be inspected more thoroughly. The CosR-walk goes one step further and investigates the relations between consecutive windows [11]. The CosR score is again a similarity score, and the CosR-walk combines this score with a sliding window approach to compare two consecutive windows. With this it is possible to determine if two legitimate windows or one legitimate and a covert window follow each other. [11], [26]. Since our focus was detection based on a single window, and we only used pure covert channel recordings instead of mixed recordings, we did not investigate these methods further as to minimize false-positives.

Related Detection Methods: In 2014, JitterBug, the MB-CTC and the TRCC were evaluated by Archibald et al. [27] using KS test, Welch's t-test, Entropy evaluation, regularity score and a shape test in conjunction with the WAND NZIX dataset [28] from July 2000, which cannot be considered as a valid example of 2022's Internet traffic characteristics anymore.

Han et al. recently analyzed the detectability of several covert timing channel variants using SVM, kNN, Naive Bayes and Logistic Regression [29], where the highest detection rates where achieved with SVM and kNN. For the *Receiver Operator Characteristic* (ROC), the authors reached AUC (*Area Under Curve*) values of 0.9727 (kNN) and 0.9452 (SVM), respectively.

Li et al. proposed another machine learning-based pipeline called *Generic and Sensitive (GAS) anomaly detection* in [13], which employs a recurrent neural network (RNN), in particular an LSTM. GAS has shown good performance on timing channel detection. They compared GAS with several other detection methods, including statistical methods (KS and regularity), entropy-based methods, and SVM.

Al-Eidi et al. proposed a new detection method called *Snap-Catch* [14]. For this approach, network traffic is first transformed into 16x16 pixel images to afterwards extract several features rooted in image processing, like mean grey value, center of mass and standard deviation of grey values. These features are then used to train several machine learning models. The authors evaluated their feature set against other approaches such as CCE, regularity and entropy. In their tests, SnapCatch outperforms the other evaluated methods, resulting in almost perfect accuracy when detecting simple covert timing channels.

Finally, Wu et al. tested the detectability of different covert timing channels using ϵ-similarity, KS test, Entropy and Corrected Conditional Entropy tests as well as regularity metric in [30]. This is the only current work that evaluates the ϵ-similarity, and the authors reported that two of the tested channels where detectable with an accuracy of 98% and 100%, while JitterBug and TRCC were not detectable. In comparison to Wu et al. we show that a much simpler covert timing channel can already significantly decrease the performance of the ϵ-similarity while our channel can moreover decrease the detectability of the compressibility score, thus, showing key weaknesses in both detection methods.

We conclude that none of the previous works analyzed the compressibility score against sophisticated covert timing channels and only one work analyzed the ϵ-similarity for such a scenario. Given the high number of citations, we decided to investigate the ϵ-similarity and the compressibility score in detail and show that they cannot handle ϵ-κlibur, which – in contrast to previous works – does not rely on the modulation of pre-recorded or complex traffic generation frameworks while providing a high bitrate.

Derived Heuristics for Alternative Covert Channels: Moreover, the two detection approaches have been adapted by other researchers in order to work with different covert channels. Zillien et al. modified both detection approaches in [23] to work with a covert channel that uses the timing of artificial TCP retransmissions to transmit hidden data, similarly to the IAT covert channel. To adapt the detection methods to this new covert channel, the authors involved the distance between succeeding TCP sequence numbers. This distance measure is then used as the input for the ϵ-similarity and the compressibility score. The authors found that the ϵ-similarity is a promising approach to detect the retransmission covert channel, while the compressibility score alone did not perform well enough but could be used as a feature for a more sophisticated detection approach.

Fu et al. created covert channels in IaaS environments and applied the ϵ-similarity for their detection and reported statisfying results [31].

Wendzel et al. modified the compressibility score, the ϵ-similarity as well as the so-called regularity metric to detect covert channels that modulate the sizes of network packets to transfer a secret message [32].

Accuracy, precision, and recall relied on the covert channel's configuration, which implies that these heuristics are suitable to detect highly specific covert channels but remain fragile to disturbances. Further, Mileva et al. used a method based on the compressibility score to detect two covert channels using the MQTT 5.0 IoT protocol — the results have shown that for one of the covert channels, the applied coding influenced the detectability significantly while the other covert channel was well-detectable with different configurations of their testbed [33].

As can be seen, ϵ-similarity and compressibility score are actively used to detect covert channels of different types but tests on their functioning on sophisticated timing channels are lacking, rendering these popular methods not well-understood.

IV. Design of ϵ-κlibur

The aim for our research is to find ways to manipulate the IATs of the covert channels in such a way that we minimize the detectability using ϵ-similarity and compressibility score. We further change the statistically observable behavior of the cover channel without compromising the bandwidth or the reliability of the transmission. This means that we try to create a high-bandwidth covert channel that both methods cannot detect. As both detection methods try to find some sort of regularity or pattern in the IATs of network traffic to detect the covert channels, we tried to break this regularity by changing the timing behavior of the covert channel. First, to have a numerical measure on how much our changes would impact the reliability of the covert channels, we developed an *impact score* that would tell us how many symbols would be unintentionally changed by modifying the delay too much.

For this impact score (Algorithm 1), we compare the original delays d_i that the covert channel would normally use, and the modified delays d'_i. We determine if we would accidentally move a delay from one side of the decoding threshold t to the other (i.e., whether the symbol interpretation would be flipped).

With this measure, we can quantify how strongly we influence the covert channel's decoding quality. Ideally, the impact score I should remain at 0 to prevent introducing any decoding errors,

Algorithm 1: Impact Score.

```
Data: List of pairs of original covert channel delays
      and corresponding modified delays $D$;
      Decoding threshold $t$
Result: Impact score $I$
Error counter $E \leftarrow 0$;
foreach $(d_i, d'_i)$ in $D$ do
    if $(d_i \leq t$ and $d'_i > t)$ or $(d_i > t$ and $d'_i \leq t)$ then
        $E \leftarrow E + 1$;
    end
end
$I \leftarrow E/|D|$;
return $I$;
```

Algorithm 2: Fuzziness Injection.

```
Data: Delay $d$(sec), Covert channel base delay $\tau$(sec)
Result: Modified delay $d'$(sec)
$threshold \leftarrow (3\tau)/2$;
if $d \leq threshold$ then
    $d'$ is drawn from the normal distribution
     $\mathcal{N}(0, (threshold/7))$;
    $d' \leftarrow |d'|$;
    if $d' > threshold$ then
        $d' \leftarrow threshold$;
    end
else
    $d'$ is drawn from
     $\{threshold \ldots (2.4 \cdot \tau), stepsize = 0.001\}$;
end
return $d'$;
```

but depending on the networking environment and whether an error-correcting code is being used, smaller values of I might still be tolerable. In our proof-of-concept implementation, we did not apply an error-correcting code.

A. Injection of Fuzziness

Our approach to break up the temporal structure of the covert channels is the systematic injection of fuzziness. The basic idea of this approach is to make the timings of the covert channel less precise and therefore obscure the structure.

Normally, the covert sender would choose one of the delays, corresponding to the hidden symbol that is to be sent next, and then delay the next network packet by that amount. This results in the covert sender using the same delays over and over again. If the covert channel uses only two symbols, this effect is maximized, as there are only two possible delays. With more than two symbols, this effect is lessened to some extent, as more different delays are used while transmitting the hidden data. But even with several different symbols, a significant degree of structure will still remain.

With our new approach, the covert sender will no longer just choose a delay from a fixed list, instead the sender applies a post-processing function to the delay before using it in the sending process. The corresponding function is shown in Algorithm 2.

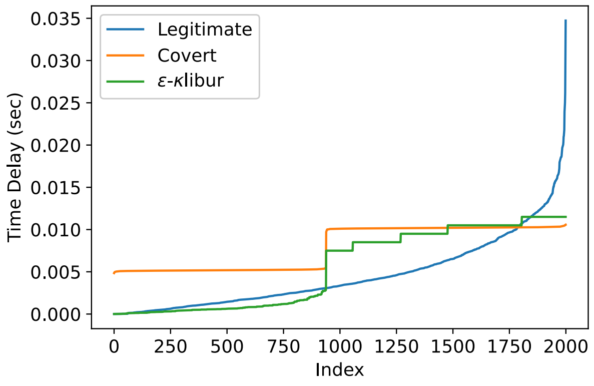


Fig. 3. Sorted IATs of the original covert channel ($\tau = 5\ ms$), ϵ-κlibur and legitimate traffic.

Fig. 3 shows the sorted IATs of regular, covert and ϵ-κlibur traffic. As visible, the curve for ϵ-κlibur has two distinct parts, one defined by the normal distribution, the other one by the stepwise function. We chose the normal distribution as it resembles the curvature of the legitimate traffic, only scaled down. For the second half of the graph, we chose a stepwise function. If we were to use another normal distribution with an offset (to ensure the separation between two symbols), we would have significantly lower λ values, as the delays are generally higher while their differences stay on the same level as before. To counter the offset, we would have to increase the scale of the distribution to unfeasibly high levels. Therefore, we chose the stepwise function. Each of these sharp steps result in a large spike in the λ values, compared to a smooth curve with the same upper and lower bounds, which will only produce low λ values. This choice of functions allowed us to push down the ϵ-similarity-score further.

The scale of $threshold/7$ for the normal distribution and the upper limit of 2.4τ in the `else` branch have shown good results for both detection methods in our empirical evaluation. Our goal was to optimize both detection values simultaneously. We therefore had to find a balance that would reduce the effectiveness of both algorithms at once without accidentally benefitting one or the other.

The threshold choice of $(3\tau)/2$ stems from the configuration of the covert channel. Since the covert channel uses a timing configuration of τ and 2τ, $(3\tau)/2$ gives us a threshold in the middle of the two values.[1]

Algorithm 2 accomplishes multiple things:

1) Different IATs are spread apart from each other.
2) IATs closer to 0 and with more zeros after the decimal point are introduced.
3) The "slope" of the sorted IATs becomes more similar to that of legitimate traffic.
4) Delays that were below the decoding threshold will still be below the threshold, delays above will still be above. In all our tests, we reached an impact score of $I = 0$, so we did not introduce any decoding errors.

[1]In our evaluation (Section V), we experiment with this threshold using different τ values.

This algorithm can also be easily adapted to covert channels with more than two symbols by adding more stages to the `if-else` block.

Covert Channel Bitrate: Depending on the τ values, ϵ-κlibur provides a different bitrate. Our fastest configuration ($\tau = 5\ ms$) achieved $\approx$185 symbols per second. With our encoding schema, this results in $\approx$185 Bits/s. The original covert channel with the same configuration achieved $\approx$127 Bits/s, so we even increased the bandwidth by introducing delays that are smaller on average.

V. Evaluation

Implementation of Covert Channels for Evaluation: In our implementation of the covert channels, we used two timings, τ and 2τ, a short and a long delay, resulting in a morse like encoding. The covert receiver monitors the delays of consecutive packets from the covert sender to then measure and decode the IATs. It is also possible to use more delays, resulting in a more complex encoding. We decided against this, as a two-symbol covert channel is the most difficult scenario for circumventing the detection heuristics as it is the easiest to detect, since the introduced structure will be the clearest (as we discussed before).

To create the recordings for the original covert channel, we used the tool CCEAP [34]. The tool offers a simple interface to create different covert channels, including IAT-based covert channels. For our tests, we used various different timing intervals to get a broader overview of the performance.

Used Traffic Recordings: The original paper as well as several later works, such as [27] (2014) and [32] (2019), used reference recordings from the NZIX II dataset [28], which was recorded in the year 2000. Since this dataset is now more than 20 years old and networking hardware has changed a lot since then, we decided to create new reference recordings. We chose four different activities that are most prevalent in the recent years: video streaming, video conferences, online gaming and file downloads. All recordings were performed on the WAN interface of a home internet gateway. In total, we recorded 4.8 GByte of reference recordings from which we extracted roughly 1,790,000 packets. We believe that this traffic mix represents the average Internet usage today more closely than the original NZIX recordings.

A. ϵ-Similarity Evaluation

The first detection method that we evaluated is the ϵ-similarity. We used the settings and thresholds from the original paper when evaluating our approach, all scores were calculated with a window size of 2,000 packets as this was also used in the original paper.

Our datasets include several different configurations for the covert channel. We used τ values of 5, 10, 20, 30, 40, 50 and $100\ ms$ for our tests.

We evaluated three different splits of covert and legitimate traffic in order to simulate realistic, best- and worst-case situations for the detection (50/50, 1/99 and 99/1 mixture of legitimate/covert channel traffic).

To quantify the performance of the detection methods, we chose AUC. The AUC generally show us how well the two datasets (legitimate and covert channel) can be separated regarding the ϵ-similarity or the compressibility score, respectively. A steeper curve and a higher AUC signal a good separation, while a flatter curve with a lower AUC signals a worse separation. We do not want to minimize the AUC too far, as values below 0.5 can be flipped above 0.5 by inverting the detection labels [35]. Therefore, the goal for ϵ-κlibur was to bring the AUC as closely to 0.5 as possible.

In Fig. 4, we compare the box plots of the ϵ-similarity for legitimate, original covert channel and ϵ-κlibur traffic. We can see that scores for the legitimate traffic are spread and their median rises with the ϵ-threshold. The ϵ-similarity scores for the original covert channel are above 0.95 for all ϵ-thresholds. Thus, it is easy to see that the original covert channel can be detected with this method, while ϵ-κlibur produces significantly lower values than the original covert channel. While the ϵ-similarity values for the ϵ-κlibur dataset are still closer to each other compared to the values of legitimate traffic, we can observe that the values blend in better with the legitimate traffic. This shows us that the basic principle of our approach works for the ϵ-similarity.

Table I provides the results for the different splits in the datasets. ϵ-similarity can easily detect all the original covert channels in various dataset splits, as all AUC values are 1.00 (rounded to 2 decimals). We can also see that the different ϵ-thresholds are all equally efficient in this situation.

Table II gives an overview of the AUC values in relation to different τ values. For the original covert channel, all values are again 1.00. With this and the results from Table I, we can observe that the delay configuration of the covert channel and the split of the dataset do not have an influence on the results of the ϵ-similarity.

The AUC values for ϵ-κlibur show a clear change in the detection performance. In Table I, we can see that our approach generally reduces the AUC across all ϵ-thresholds. We can also observe that the split of the dataset does not influence the effectiveness of our approach. Table II shows the effectiveness of the fuzziness in relation to the covert channel configuration and ϵ-threshold. We can again determine that the effectiveness of ϵ-κlibur is reduced with larger ϵ-thresholds, but even the best detection result is still only at AUC = 0.87. This would limit the usefulness of the detection heuristic in a real life scenario given that several GBit/s of flow data would need to be processed where even a small false-positive rate would render the approach impractical. Moreover does the best value (AUC = 0.87) only apply to one type of covert channel, while realistic setups have to deal with different potential covert channel configurations, which do not perform well, as already shown in Table I.

Table II shows that the detectability of ϵ-κlibur changes depending on the τ values. Specifically for higher τ values, it is harder to produce higher relative differences, as the decoding threshold forces higher IATs. This explains the worse performance for higher ϵ-thresholds with higher τ values.

Fig. 5(a) shows that we obtain a perfect detection of the unmodified covert channels, while in Fig. 5(b), we can observe

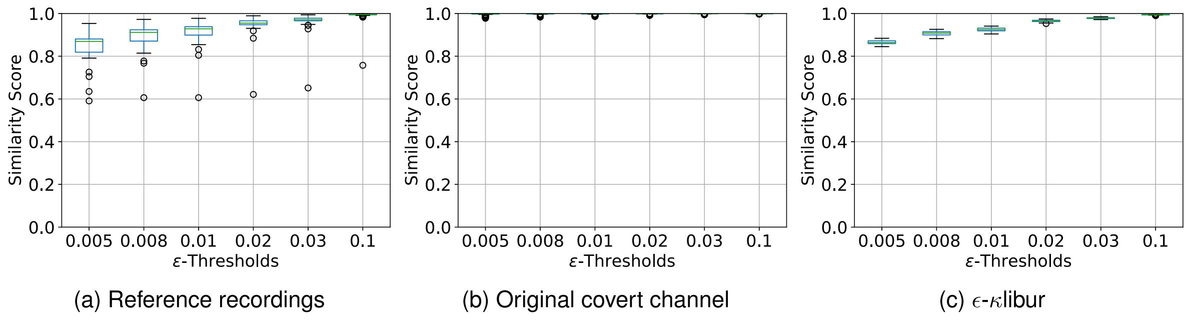


Fig. 4. ϵ-similarity score comparison (mixed traffic with all τ configurations).

TABLE I
ϵ-SIMILARITY: AUC VALUES OF MIXED TRAFFIC WITH ALL τ CONFIGURATIONS (THRESHOLDS AS GIVEN BY CABUK ET AL.)

		$\epsilon = 0.005$	$\epsilon = 0.008$	$\epsilon = 0.01$	$\epsilon = 0.02$	$\epsilon = 0.03$	$\epsilon = 0.1$
Original	50/50 Split	1.00	1.00	1.00	1.00	1.00	1.00
	99/1 Split	1.00	1.00	1.00	1.00	1.00	1.00
	1/99 Split	1.00	1.00	1.00	1.00	1.00	1.00
ϵ-κlibur	50/50 Split	0.48	0.49	0.51	0.72	0.77	0.37
	99/1 Split	0.48	0.49	0.51	0.72	0.77	0.37
	1/99 Split	0.49	0.50	0.51	0.72	0.76	0.36

Background color shows distance to 0.5, high distance: green (good detectability), low distance: red (bad detectability)

TABLE II
ϵ-SIMILARITY: AUC VALUES FOR ISOLATED COVERT CHANNEL CONFIGURATIONS (THRESHOLDS AS GIVEN BY CABUK ET AL.)

		$\epsilon = 0.005$	$\epsilon = 0.008$	$\epsilon = 0.01$	$\epsilon = 0.02$	$\epsilon = 0.03$	$\epsilon = 0.1$
Original	τ = 5 ms	1.00	1.00	1.00	1.00	1.00	1.00
	...	...	...	...	...	...	...
	τ = 100 ms	1.00	1.00	1.00	1.00	1.00	1.00
ϵ-κlibur	τ = 5 ms	0.59	0.54	0.56	0.67	0.65	0.16
	τ = 10 ms	0.62	0.53	0.55	0.62	0.64	0.36
	τ = 20 ms	0.54	0.49	0.48	0.56	0.71	0.41
	τ = 30 ms	0.41	0.47	0.46	0.70	0.84	0.47
	τ = 40 ms	0.43	0.44	0.41	0.83	0.83	0.30
	τ = 50 ms	0.40	0.41	0.46	0.84	0.87	0.36
	τ = 100 ms	0.38	0.58	0.61	0.82	0.82	0.53

that our modifications resulted not only in a reduction of the AUC to 0.48, but also did we manage to push the curve closer to the diagonal line. This means that the detection algorithm provided unfavorable scaling between true- and false-positives over large portions of the entire range, i.e., only after 60% false-positives, we could significantly gain additional true-positives without also suffering more false-positives. For our approach, this is an almost optimal result.

In Fig. 5(c) and (d), we compare the ROC curves for two different covert channel configurations. In Fig. 5(c), we can see the curves for $\tau = 5\ ms$ and in Fig. 5(d) for $\tau = 30$ ms. Both plots show the results for ϵ-κlibur.

In Fig. 5(c), the AUC is larger than in Fig. 5(d), but both have the same distance from 0.5. Both figures show a rather steep ROC curve, so if we only look at a single covert channel configuration, we have a certain threshold from which on we only gain true-positives without suffering many more false-positives.

While this technically denotes worse performance from our approach, we still believe that the performance of the detection algorithm is still not useable in this state.

In a real-world scenario, a detector would have to base its detection thresholds around all possible covert channels, so only comparing a single covert channel configuration does not give a real-world image. We therefore mostly focussed our evaluations on datasets that include multiple covert channel configurations. However, the more detection thresholds are applied simultaneously, the higher the number of cumulative false-positives.

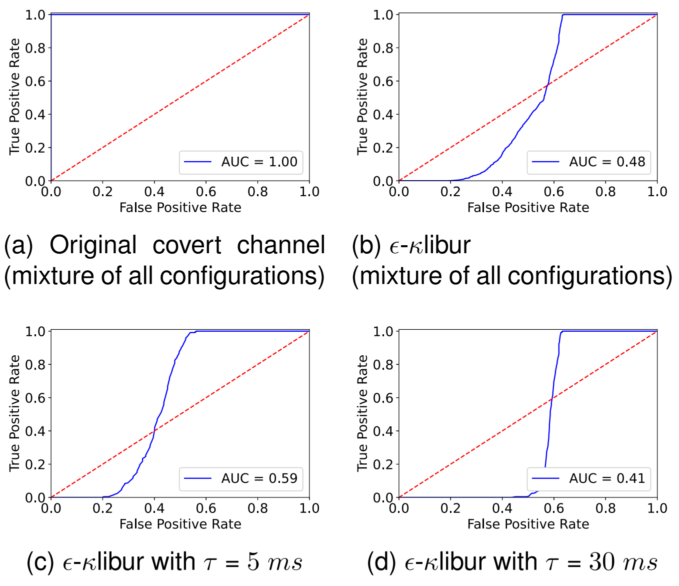


Fig. 5. ROC curve comparison for ϵ-similarity.

TABLE III
AUC - Compressibility, Depending on Split

	Original	ϵ-κlibur
50/50 Split	1.00	0.40
99/1 Split	1.00	0.39
1/99 Split	1.00	0.39

B. Compressibility Score Evaluation

The second detection heuristic that we evaluated is the compressibility score. We also used the same settings from the original paper and used a windows size of 2,000 packets. Similar to the ϵ-similarity, we applied different configurations for the covert channel delays in order to reach a broader view of the performance of our approach. We used the same configuration of τ and 2τ for short and long delays as well as the same list of τ values of 5, 10, 20, 30, 40, 50 and 100 ms.

Similar to the ϵ-similarity, our evaluation was based on the AUC of a ROC curve as a measure of how well the detection algorithm works on the original covert channel and ϵ-κlibur. Our goal was again an AUC of 0.5 with a slope as close to 1 as possible.

Fig. 6 compares the histograms of the compressibility scores of legitimate, covert and ϵ-κlibur recordings. Similar to the ϵ-similarity, we observe a clear difference between the covert channel and ϵ-κlibur. We can easily notice that compressibility values are significantly lower for ϵ-κlibur and blend in well with the legitimate values.

This shows that the basic idea of the fuzziness injection also works for the compressibility score. In Fig. 6, we can moreover determine a different distribution of the compressibility scores for legitimate, covert and ϵ-κlibur traffic. ϵ-κlibur's distribution of κ values overlaps significantly with legitimate traffic.

TABLE IV
AUC - Compressibility, Depending on τ

	Original	ϵ-κlibur
τ = 5 ms	1.00	0.68
τ = 10 ms	1.00	0.54
τ = 20 ms	1.00	0.51
τ = 30 ms	1.00	0.31
τ = 40 ms	1.00	0.31
τ = 50 ms	1.00	0.25
τ = 100 ms	1.00	0.16

Table III lists the AUC values for the compressibility score for different splits of the dataset. We can see that the compressibility score can also detect all original covert channels for each tested split. All AUC values are again 1.00 (rounded to 2 decimals), so the algorithm can perfectly distinguish between covert channel and legitimate traffic.

Table IV presents the detection performance in relation to the different delay configurations. We can again observe that the compressibility score can detect all original covert channels, no matter their configuration. Thus, similar to the ϵ-similarity, delay configurations and dataset splits have no impact on the raw performance of the compressibility score.

If we look at the AUC values for ϵ-κlibur, we can again notice a clear difference. In Table III, it is visible that the AUC values are lower and around 0.40. We can thus conclude that our approach also works for the compressibility score.

Table IV demonstrates that our approach works for all τ configurations of the covert channel. The effect of the fuzziness depends on the delay configuration. Some values are below 0.5 while others are above 0.5 and – if we combine all recordings in a dataset – we obtain AUC values around 0.4.

This scaling behavior can be explained by the second part of the fuzziness injection algorithm. Since the upper bound for the random delays scales with the value of τ, we get a larger spread in possible values, which in turn result in more different elements in the string (see Section II.B.2) which leads to a lower compressibility. Even if we take the worst performance (from our viewpoint), which is an AUC of 0.16, and flip the labels, we would end up with an AUC of 0.84. But this result is limited to channels with $\tau = 100\ ms$ and is not well-useable in a real-world scenario, as the detector could not optimize its thresholds only for a single covert channel configuration but would rather have to apply the detection to *all* possible covert channels (scenario of Table III). Even if we focus on only this single worst-case configuration, we still would suffer from a high false-positive rate ($>$13%) if we want to achieve a true-positive rate of over 82%.

In Fig. 7, we compare the ROC curves for the original covert channel and ϵ-κlibur. The figures underpin the perfect detection for the original covert channel in contrast to the results for ϵ-κlibur. Not only did we reduce the AUC to around 0.4, we also managed to achieve an almost diagonal line. That means the detector has bad scaling between true- and false-positives throughout the entire range.

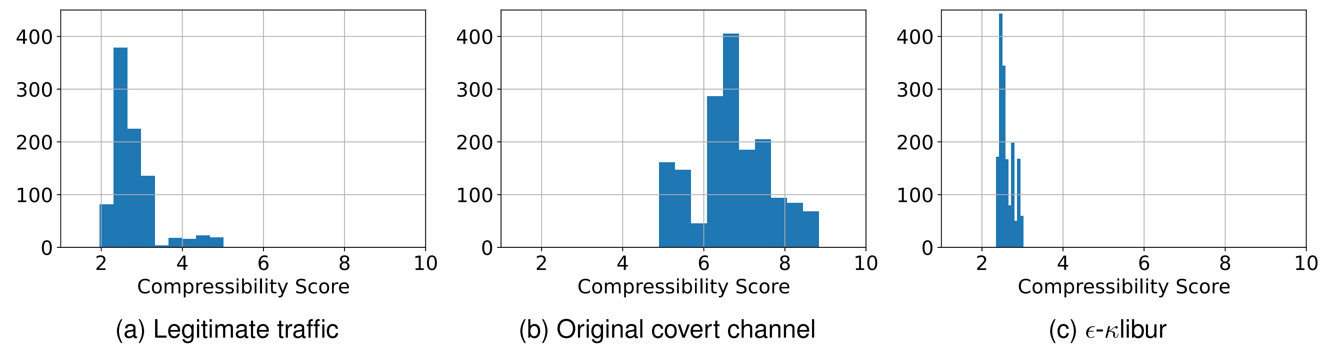


Fig. 6. Compressibility score comparison.

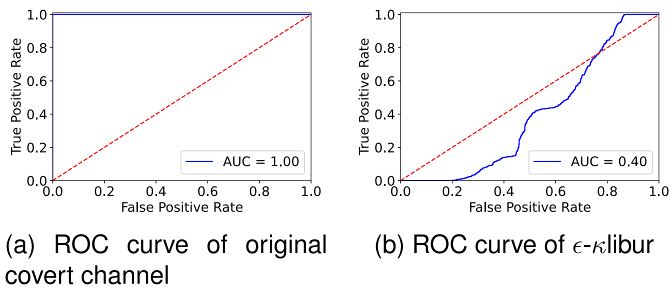


Fig. 7. ROC curve comparison for the compressibility score.

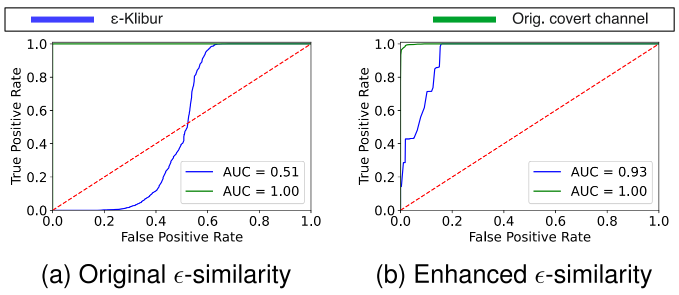


Fig. 8. ROC curve comparison for the original ϵ-similarity and the enhanced ϵ-similarity (plot shows $\epsilon = 0.01$).

VI. Enhanced Detection Approach Based on ϵ-Similarity

In this section, we discuss an adaption of the ϵ-similarity that can be used to counter ϵ-κlibur. Our adaption follows the original approach closely but has one significant modification. Instead of looking at an entire window at once, we first sort the IATs and then divide the window into three equal subwindows. Thus, with an original window size of 2,000, the first and second subwindow will contain 667 IATs and the third subwindow will contain 666 IATs.

As both symbols of the covert channel occur with an equal probability of $p \approx 0.5$, we will have only low IATs in the first subwindow, a mix of low and high in the second and only high values in the last subwindow. With legitimate traffic, we see a steep incline in the last subwindow (see Fig. 3). The original covert channel is flat, and our ϵ-κlibur has several steps in this subwindow. ϵ-κlibur has two constraints in the last subwindow. First, there is a hard lower bound, as no IATs below the decoding threshold can reside in this window. Theoretically, we could use arbitrarily large IATs, but this would not be feasible in a real life scenario. Therefore, the IATs in this third subwindow are constrained between two bounds, which in turn leads to lower relative differences in this subwindow. Therefore, we can use this third subwindow for the new detection measure by applying the original ϵ-similarity to it. We evaluated our new detection approach against the original covert channel and ϵ-κlibur.

Fig. 8 shows the detection results for the two different covert channels, the original one and our ϵ-κlibur. The results show that

TABLE V
AUC – TRRC, ϵ-Similarity versus Enhanced ϵ-Similarity

	Recording 1 (Gaming)		Recording 2 (MS Teams)	
	ϵ-similarity	enhanced	ϵ-similarity	enhanced
ϵ = 0.005	0.63	0.88	0.98	0.87
ϵ = 0.008	0.47	0.85	0.98	0.82
ϵ = 0.01	0.43	0.85	0.94	0.81
ϵ = 0.02	0.51	0.83	0.51	0.74
ϵ = 0.03	0.50	0.80	0.06	0.61
ϵ = 0.1	0.14	0.83	0.00	0.28

we were able to increase the detection performance for our ϵ-κlibur without hurting the detection performance for the original covert channel significantly.

TRCC Evaluation: We decided to perform an additional evaluation of our enhanced ϵ-similarity heuristic to analyze whether it is also able to detect the TRCC (see Section III.A) better than the original approach. TRCC depends on legitimate traffic. For this reason, we tested TRCC with two different reference recordings and found drastically different detection results for the two recordings.

Table V(a) shows the results for our first reference recording (which is a recording of an online gaming session). With this recording, the TRCC was (except for the last ϵ-threshold) hard to detect for the original ϵ-similarity but our enhanced detection approach delivered significantly improved results with AUC values between 0.80 and 0.88, despite being not optimally suited for real-world scenarios.

With the second reference recording (which is a recording of a Microsoft Teams meeting), shown in Table V(b), the TRRC was well detectable by the original ϵ-similarity with the first three ϵ-thresholds (AUC $> 90\%$). $\epsilon = 0.02$ delivered unusable detection performance, while the last two delivered a good performance again, although with flipped labels (AUC $< 10\%$). Our enhanced ϵ-similarity delivered, in some cases, worse and in other cases better performance compared to the original ϵ-similarity. Generally, the enhanced ϵ-similarity delivered a more consistent detection performance.

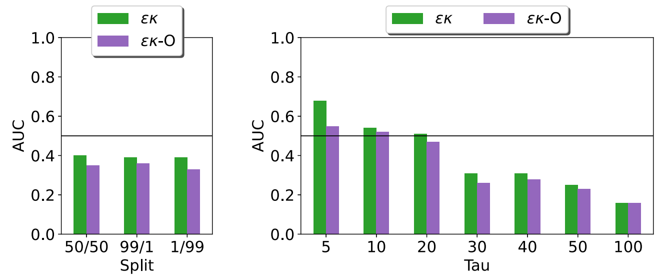
The strong fluctuations in detection performance regarding the different ϵ-thresholds and the reference recordings lead to inconclusive results. Therefore, we believe that the detection performance is more dependent on the reference recording than anything else, and thus one would need to conduct further research focusing solely on the parameters and statistics of the reference recordings to sufficiently evaluate the TRCC.

VII. Evaluation of ϵ-κlibur With Sophisticated Detection Methods

So far we have shown that ϵ-κlibur can circumvent both, the compressibility score and the ϵ-similarity, and that the enhanced ϵ-similarity method outperforms both classical heuristics. While the main focus of our work was to demonstrate weaknesses and exploitability of these classical heuristics, we additionally evaluate ϵ-κlibur against two recent machine learning-based detection methods: GAS [13] and SnapCatch [14] as introduced in Section III.B. Since ϵ-κlibur was solely tailored to circumvent the compressibility score and the ϵ-similarity, we slightly adjusted our method by adding an additional outlier timing to the high inter-arrival signal of ϵ-κlibur (the `else`-branch of Algorithm 2 is slightly altered for this purpose), which we call ϵ-κlibur-O. The idea here is to stretch the overall distribution of the timings further apart. Regular traffic showed a steep increase in delays (see Fig. 3) which we wanted to imitate with these outliers. We used an outlier timing of 10τ.

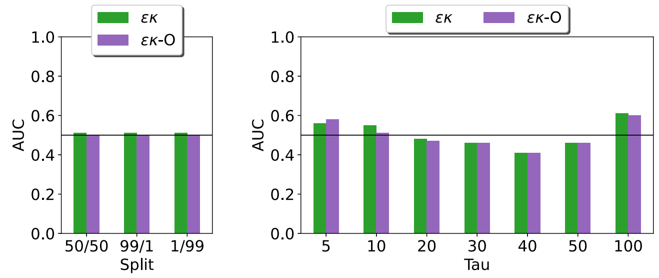
A. *Performance of ϵ-κlibur-O Against Compressibility and ϵ-Similarity*

Our evaluation has shown that ϵ-κlibur-O yields similar results as ϵ-κlibur when the compressibility score or the ϵ-similarity are used. For this evaluation, we again used the AUC value as a performance metric. As a high AUC value shows a good detection performance and a low AUC value ($\ll$0.5) also leads to a good detection by flipping the labels, we focused on the distance of the AUC score to 0.5. Fig. 9 shows the AUC scores for the compressibility of ϵ-κlibur and ϵ-κlibur-O based on a) dataset split (as explained in Section V.A) and b) covert channel configuration. We can see that for most configurations there is no significant deviation for ϵ-κlibur-O compared to ϵ-κlibur. For the configuration with $\tau = 5$ ms and 10 ms, we can even observe a better performance for ϵ-κlibur-O, as the AUC value is closer to 0.5, for $\tau = 20$ to 50 we see a slight performance decrease.



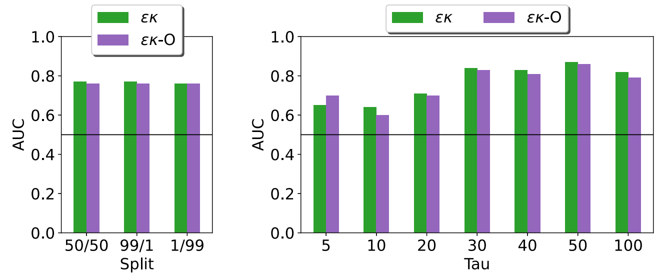
(a) Based on Dataset Split (b) Based on CC Configuration

Fig. 9. AUC comparison for Compressibility Score, ϵ-κlibur versus ϵ-κlibur-O.



(a) Based on Dataset Split (b) Based on CC Configuration

Fig. 10. AUC comparison for ϵ-similarity $\epsilon = 0.001$, ϵ-κlibur versus ϵ-κlibur-0.



(a) Based on Dataset Split (b) Based on CC Configuration

Fig. 11. AUC comparison for ϵ-similarity $\epsilon = 0.003$, ϵ-κlibur versus ϵ-κlibur-0.

Figs. 10 ($\epsilon = 0.001$) and Fig. 11 ($\epsilon = 0.003$) show the results for the AUC values of the ϵ-similarity. Similarly to the compressibility score, some configurations suffered slightly in performance while others slightly gained. For most configurations of the compressibility and ϵ-similarity, the changes in AUC are below 0.1. So there is no clear trend across all configurations that points towards ϵ-κlibur-O being significantly easier or harder to detect compared to ϵ-κlibur, when faced with compressibility or ϵ-similarity.

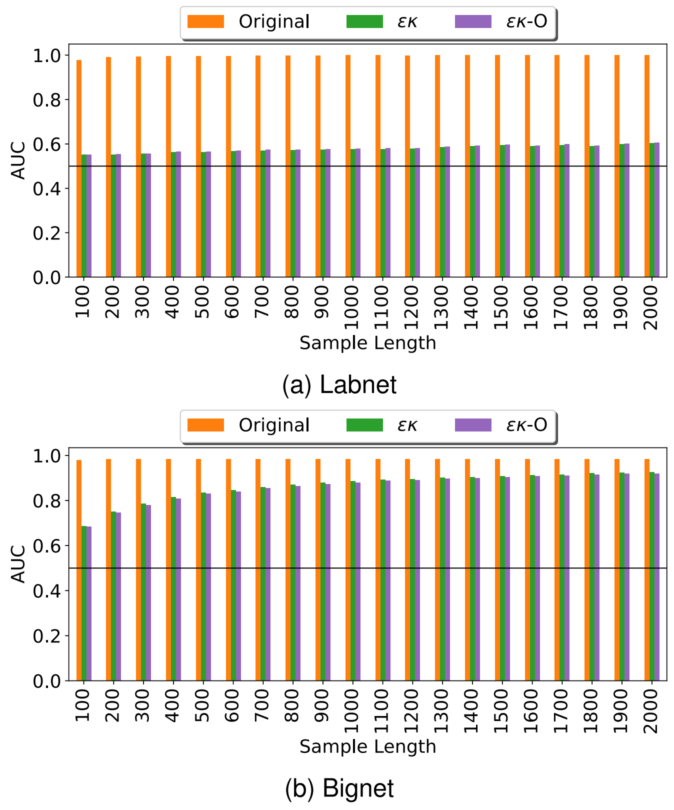


Fig. 12. AUC comparison for GAS, orig. CC versus ϵ-κlibur versus ϵ-κlibur-O.

B. Performance of ϵ-κlibur-O Against the Enhanced ϵ-Similarity

We also evaluated ϵ-κlibur-O against our enhanced version of the ϵ-similarity. We ran the same evaluations as before and used the AUC as performance metric. In our experiment, we found that the AUC remained nearly unchanged for most configurations and only showed a noticeable degradation at $\epsilon = 0.1$ (0.1 AUC decrease). Therefore, we can conclude that our enhanced detection performs equally well on ϵ-κlibur-O as on ϵ-κlibur.

C. GAS

To evaluate the *GAS* detection approach, we used the code and models provided by the original authors [13] and conducted three tests. We first tested the original covert channel, then ϵ-κlibur and ϵ-κlibur-O. Fig. 12 shows the results of our tests. We could reproduce the results of the original paper, as GAS also achieved good detection results with AUC values of 0.97 and above for the original covert channel. When faced with ϵ-κlibur and ϵ-κlibur-O on the other hand, we could observe a significant performance degradation. In parallel to the original paper, we evaluated the performance with the "Labnet" and "Bignet" setups and also with varying window sizes. In the "Labnet" setup, ϵ-κlibur and ϵ-κlibur-O degraded the detection performance significantly and pushed the AUC for all window sizes down to values between 0.55 and 0.6. The performance in the "Bignet" setup was better but ϵ-κlibur and ϵ-κlibur-O still showed a sizeable impact with AUC values ranging from 0.68 to 0.92. We thus conclude that both, ϵ-κlibur and ϵ-κlibur-O, significantly impact the performance of GAS, especially for smaller sample lengths.

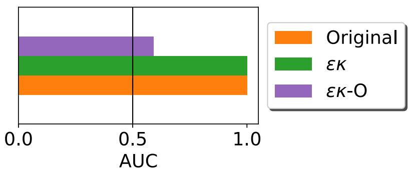


Fig. 13. AUC comparison for SnapCatch, orig. CC versus ϵ-κlibur versus ϵ-κlibur-O.

D. SnapCatch

Similar to GAS, we evaluated *SnapCatch* [14] with three datasets. The original covert channel, ϵ-κlibur and ϵ-κlibur-O. We re-implemented the feature extraction of SnapCatch and used the resulting features to train a SVM model. In each case, we trained the model with the original covert channel and tested the resulting model against the other dataset. Thus, we assume a defender with knowledge about the classical covert channel but without knowledge of ϵ-κlibur and ϵ-κlibur-O. Fig. 13 shows the results of these experiments. We can observe that SnapCatch achieved an outstanding detection of the original covert channel with an AUC of 1.0. Even ϵ-κlibur showed no significant degradation in detection performance, also resulting in an AUC of 1.0. ϵ-κlibur-O on the other hand was successful in decreasing the performance of SnapCatch and resulted in an AUC of 0.59. The data processing pipeline of SnapCatch includes a step in which the timings of each window are normalized and mapped to the range between 0 and 255. The timing spread of regular traffic resulted in images that are generally dark with only a few bright pixels. While the original ϵ-κlibur resulted in images with around 50% bright and 50% dark pixels. The outlier of ϵ-κlibur-O influenced the normalization step and therefore the images had again a few bright pixels in a generally darker image, which helped to bring the pixel values closer to those of regular traffic.

VIII. Conclusion

First, we have shown that the two highly cited covert channel detection metrics ϵ-similarity and compressibility score can be defeated with a simple covert channel that we call ϵ-κlibur. In comparison to previous attempts, ϵ-κlibur is easier to construct and requires no pre-recorded traffic while providing an non-degraded bitrate. We second introduced an enhanced ϵ-similarity that is able to detect both, the original timing covert channel as well as ϵ-κlibur. Third, we evaluated ϵ-κlibur against two more recent approaches: ϵ-κlibur can defeat *GAS* and a slight variation of ϵ-κlibur called ϵ-κlibur-O can also significantly lower the performance of *SnapCatch*. We conclude that the enhanced ϵ-similarity heuristic can compete and partially outperform the most recent ML-based methods. However, in comparison to the other methods, the enhanced ϵ-similarity was not evaluated against other sophisticated timing channels, such as MB-CTC or JitterBug.

In future work, we plan to extend our work to additional detection algorithms as well as covert storage channels. We also plan to evaluate the enhanced ϵ-similarity against other timing channels.

Notes on Replicability. We made our implementation of ϵ-κlibur available: https://github.com/NIoSaT/epskalibur.

References

[1] L. Caviglione et al., "Tight arms race: Overview of current malware threats and trends in their detection," *IEEE Access*, vol. 9, pp. 5371–5396, 2021.
[2] J. Hielscher, K. Lamshöft, C. Krätzer, and J. Dittmann, "A systematic analysis of covert channels in the network time protocol," in *Proc. 16th Int. Conf. Availability Rel. Secur.*, ACM, 2021, Art. no. 69.
[3] W. Mazurczyk and L. Caviglione, "Steganography in modern smartphones and mitigation techniques," *IEEE Commun. Surv. Tuts.*, vol. 17, no. 1, pp. 334–357, First Quarter 2015.
[4] M. Mehic, J. Slachta, and M. Voznak, "Whispering through DDoS attack," *Perspectives Sci.*, vol. 7, pp. 95–100, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2213020915000610
[5] E. Wustrow, C. M. Swanson, and J. A. Halderman, "TapDance: End-to-Middle anticensorship without flow blocking," in *Proc. 23rd USENIX Secur. Symp.*, USENIX Assoc., 2014, pp. 159–174.
[6] G. Shah, A. Molina, and M. Blaze, "Keyboards and covert channels," in *Proc. 15th USENIX Secur. Symp.*, 2006, pp. 59–75.
[7] S. Gianvecchio, H. Wang, D. Wijesekera, and S. Jajodia, "Model-based covert timing channels: Automated modeling and evasion," in *Proc. Int. Workshop Recent Adv. Intrusion Detection*, Springer, 2008, pp. 211–230.
[8] R. J. Walls, K. Kothari, and M. Wright, "Liquid: A detection-resistant covert timing channel based on IPD shaping," *Comput. Netw.*, vol. 55, no. 6, pp. 1217–1228, 2011.
[9] R. Archibald and D. Ghosal, "Design and analysis of a model-based covert timing channel for Skype traffic," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2015, pp. 236–244.
[10] S. Cabuk, C. E. Brodley, and C. Shields, "IP covert timing channels: Design and detection," in *Proc. 11th ACM Conf. Comput. Commun. Secur.*, 2004, pp. 178–187.
[11] S. Cabuk, "Network covert channels: Design, analysis, detection, and elimination," Ph.D. dissertation, Center Educ. Res. Inf. Assurance Secur., Purdue Univ., West Lafayette, IN, USA, 2006.
[12] S. Cabuk, C. E. Brodley, and C. Shields, "IP covert channel detection," *ACM Trans. Inf. Syst. Secur.*, vol. 12, no. 4, pp. 1–29, 2009.
[13] H. Li, T. Song, and Y. Yang, "Generic and sensitive anomaly detection of network covert timing channels," *IEEE Trans. Dependable Secure Comput.*, 2022, doi: 10.1109/TDSC.2022.3207573.
[14] S. Al-Eidi, O. Darwish, Y. Chen, and G. Husari, "SnapCatch: Automatic detection of covert timing channels using image processing and machine learning," *IEEE Access*, vol. 9, pp. 177–191, 2021.
[15] S. Wendzel, S. Zander, B. Fechner, and C. Herdin, "Pattern-based survey and categorization of network covert channel techniques," *ACM Comput. Surv.*, vol. 47, no. 3, Apr. 2015, Art. no. 50. [Online]. Available: https://doi.org/10.1145/2684195
[16] S. Wendzel et al., "A generic taxonomy for steganography methods," *TechRxiv*, Jul. 2022. [Online]. Available: https://doi.org/10.36227/techrxiv.20215373
[17] M. Li and P. Vitanyi, *An Introduction to Kolmogorov Complexity and its Applications: Preface*, 1st ed., Berlin, Germany: Springer, 1997.
[18] S. Zander, G. Armitage, and P. Branch, "Stealthier inter-packet timing covert channels," in *Proc. Int. Conf. Res. Netw.*, Springer, 2011, pp. 458–470.
[19] K. Lamshöft, T. Neubert, J. Hielscher, C. Vielhauer, and J. Dittmann, "Knock, knock, log: Threat analysis, detection & mitigation of covert channels in syslog using port scans as cover," *Forensic Sci. Int. Digit. Investigation*, vol. 40, 2022, Art. no. 301335.
[20] N. Kiyavash, A. Houmansadr, and N. Borisov, "Multi-flow attacks against network flow watermarking schemes," in *Proc. 17th USENIX Secur. Symp.*, USENIX Assoc., 2008, pp. 307–320.
[21] A. Houmansadr, N. Kiyavash, and N. Borisov, "RAINBOW: A robust and invisible non-blind watermark for network flows," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, The Internet Society, 2009.
[22] C. Kraetzer, J. Dittmann, A. Lang, and T. Kuehne, "WLAN steganography: A first practical review," in *Proc. 8th Workshop Multimedia Secur.*, 2006, pp. 17–22.
[23] S. Zillien and S. Wendzel, "Detection of covert channels in TCP retransmissions," in *Proc. Nordic Conf. Secure IT Syst.*, N. Gruschka, Ed. Cham: Springer, 2018, pp. 203–218.
[24] X. Li, Y. Zhang, F. Chong, and B. Zhao, "A covert channel analysis of a real switch," Dept. Comput. Sci., Univ. California, Santa Barbara, CA, USA, 2011.
[25] C. Liang, T. Baker, Y. Li, R. Nawaz, and Y.-A. Tan, "Building covert timing channel of the IoT-enabled MTS based on multi-stage verification," *IEEE Trans. Intell. Transp. Syst.*, to be published, doi: 10.1109/TITS.2021.3118853.
[26] A. K. Biswas, D. Ghosal, and S. Nagaraja, "A survey of timing channels and countermeasures," *ACM Comput. Surv.*, vol. 50, no. 1, pp. 1–39, 2017.
[27] R. Archibald and D. Ghosal, "A comparative analysis of detection metrics for covert timing channels," *Comput. Secur.*, vol. 45, pp. 284–292, 2014.
[28] WAND, "NZIX II traces," Univ. Waikato, Hamilton, New Zealand, 2000. [Online]. Available: https://wand.net.nz/wits/nzix/2/
[29] J. Han, C. Huang, F. Shi, and J. Liu, "Covert timing channel detection method based on time interval and payload length analysis," *Comput. Secur.*, vol. 97, 2020, Art. no. 101952. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404820302285
[30] S. Wu, Y. Chen, H. Tian, and C. Sun, "Detection of covert timing channel based on time series symbolization," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 2372–2382, Oct. 13, 2021.
[31] X. Fu, R. Yang, X. Du, B. Luo, and M. Guizani, "Timing channel in IaaS: How to identify and investigate," *IEEE Access*, vol. 7, pp. 1–11, 2019.
[32] S. Wendzel, F. Link, D. Eller, and W. Mazurczyk, "Detection of size modulation covert channels using countermeasure variation," *J. Universal Comput. Sci.*, vol. 25, no. 11, pp. 1396–1416, 2019. [Online]. Available: https://doi.org/10.3217/jucs-025--11-1396
[33] A. Mileva, A. Velinov, L. Hartmann, S. Wendzel, and W. Mazurczyk, "Comprehensive analysis of MQTT 5.0 susceptibility to network covert channels," *Comput. Secur.*, vol. 104, 2021, Art. no. 102207.
[34] S. Wendzel and W. Mazurczyk, "POSTER: An educational network protocol for covert channel analysis using patterns," in *Proc. ACM Conf. Comput. Commun. Secur.*, New York, NY, USA: ACM, 2016, pp. 1739–1741. [Online]. Available: https://doi.org/10.1145/2976749.2989037
[35] P. A. Flach and S. Wu, "Repairing concavities in ROC curves," in *Proc. 19th Int. Joint Conf. Artif. Intell.*, Morgan Kaufmann Publishers Inc., 2005, pp. 702–707.

Sebastian Zillien received the BSc and MSc degrees from Hochschule Worms, Germany, where he is currently working toward the PhD degree and as a researcher with the Center for Technology and Transfer (ZTT). He works for the project SIVERT (*Secure & Intelligent Visualization- & Real-time Reconstruction Methods for pCT*, https://sivert.info). His research interests include network & protocol security, covert channels, steganography, and reliability. He has worked on multiple security-related research projects and has had publications, among others, at IFIP SEC, NordSec, and in the *Journal of Universal Computer Science*.

Steffen Wendzel (Member, IEEE) received the PhD and habilitation degrees from FernUniversität, Hagen, Germany, in 2013 and 2020, respectively. He is a professor of information security and computer networks with Hochschule Worms, Germany, where he is also the scientific director of the Center for Technology and Transfer (ZTT). In addition, he is a lecturer with the Faculty of Mathematics & Computer Science, FernUniversität in Hagen, Germany. Before joining Hochschule Worms, he led a Smart Building Security Research Team, Fraunhofer FKIE, Bonn, Germany. He (co-)authored more than 170 publications. Steffen served as a guest editor for major journals, such as the *IEEE Transactions on Industrial Informatics*, *Elsevier Future Generation Computer Systems*, and *IEEE Security & Privacy*. His research focus is on covert channels, Internet censorship, scientific terminology, and IoT security. Website: https://www.wendzel.de.