Investigating the efficiency of detection methods against network covert timing channels

J.K. Faber

Master Thesis
Computer Science

Faculty of Electrical Engineering,
Mathematics & Computer Science
Cyber Security Group

TU Delft

Investigating the efficiency of detection methods against network covert timing channels

by

J.K. Faber

to obtain the degree of Master of Science in Computer Science
Software Technology Track with a 4TU specialization in Cyber Security
at the Delft University of Technology,
to be defended publicly on Thursday January 28, 2021 at 9:00 AM.

Student number:	4189736	
Project duration:	May 2, 2019 – January 28, 2021	
Thesis committee:	Prof. dr. ir. R. L. Lagendijk,	TU Delft, chair
	Dr. A. Zarras,	TU Delft, supervisor
	Dr. S. Roos,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

Network covert timing channels are techniques to covertly transmit information over computer networks, by utilizing the time between subsequent network packets. Previous work on the detection of the various techniques has introduced numerous new methods, with high reported success. From these previous works we have noticed that there is little confirmation on these results in subsequent works, as well as there being a lack of an overview for the efficiency of each method. Next to this, we have found that many works use data in their experiments that may not be representative of real network scenarios.

In this thesis we attempt to remedy this lack of information, by performing a broad performance evaluation on the currently existing singular detection metrics. This performance evaluation was done on a total of 18 different detection methods, applied to the 8 most prevalent covert timing channels. For the underlying network data, we gathered SSH and HTTPS traffic from the TU Delft, and applied varying amounts of simulated network jitter to them.

From the resulting evaluations we find that there are cases where the detection methods do perform similarly to what has been shown in previous work, but we also find those that have a large difference in performance. Further, we discuss possible strengths and weaknesses of each of the detection methods, based on their performance, and in some cases how this performance might be improved. Using the (simulated) network scenarios we show the effects that jitter and different traffic types can have on each of the detection methods, and also find those that are resilient to network effects. Finally, we combine the full experimental performance evaluations into a comprehensive overview, for each combination of detection method and covert channel technique.

We find that the current detection methods are likely not sufficient to be reliably applied in a realistic network setting, and more work needs to be done in this field to reach that point. The overview and discussions we have provided can then serve as a basis for future research, to give an indication of where performance needs to be improved.

Preface

The completion of my master's degree has been a long journey, which now reaches its end with this thesis, and brings the start of a new phase in my life. There have been many obstacles along the way, not least of which an ongoing pandemic. At the start I did not know anything of the subject, but now looking back I believe I have a firm grasp on the concepts in this field of research. In the end I am proud to present what has been achieved through this thesis.

First, I would like to thank Apostolis Zarras for taking over the project on such short notice, and providing valuable feedback on finishing the thesis. It was nice hearing the appreciation for the work I put into the thesis, as well as getting the greenlight to finally graduate. Next, I also want to thank Sicco Verwer for guiding myself and other students towards the end of our theses, when we needed it most. Further, I would like to thank the rest of my thesis committee, Inald Lagendijk and Stefanie Roos, for reading my thesis and attending my presentation. In addition, I want to thank Christian Doerr for providing data and helping with the start of the thesis.

Finally, I want to thank my family and friends for their support during the project, and my education in general. In particular I am very grateful to my parents, Gunnar and Lieke, for helping me to keep going, at points in the thesis where there seemed to be no end in sight. Even though the subject is unfamiliar to them, they were also there to listen to every little detail and complaint, which helped a lot with my writing. Special thanks to Timo, Cathrien, friends, and clubmates, for being there to support me.

J.K. Faber
Den Haag, January 2021

Contents

1 Introduction **1**

2 Background and related work **3**
2.1 Covert timing channels . . . 4
2.1.1 Simple Covert Channel . . . 4
2.1.2 On-Off Covert Timing Channel . . . 4
2.1.3 Delayed Packet One Indicator . . . 5
2.1.4 JitterBug . . . 6
2.1.5 Model Based Covert Timing Channel . . . 7
2.1.6 Repeat Model Based Covert Timing Channel . . . 8
2.1.7 Time Replay Covert Timing Channel . . . 8
2.1.8 L-Bits-to-N-Packets . . . 8
2.2 Detection methods . . . 9
2.2.1 ε-Similarity, Regularity, and Compressibility . . . 9
2.2.2 Arimoto-Blahut, and Mean IPD . . . 10
2.2.3 Kolmogorov-Smirnov, Entropy, and Corrected Conditional Entropy . . . 10
2.2.4 Support Vector Machine . . . 10
2.2.5 Kullback-Leibler divergence, Skew, Kurtosis, and Welch's t-test . . . 11
2.2.6 Spearman-Rho, Wilcoxon Signed-Rank, and Mann-Whitney-Wilcoxon . . . 11
2.2.7 Summary . . . 12

3 Experimental evaluation **13**
3.1 Dataset . . . 13
3.2 Detection methodology . . . 14
3.3 CTC settings . . . 15
3.4 Detection method settings . . . 16

4 Results **19**
4.1 Entropy, Kullback-Leibler, and Corrected Conditional Entropy . . . 19
4.2 Mean IPD . . . 24
4.3 ε-Similarity . . . 25
4.4 Regularity . . . 27
4.5 Compressibility . . . 28
4.6 Welch's t-test . . . 31
4.7 Skew and Kurtosis . . . 33
4.8 Kolmogorov-Smirnov . . . 34
4.9 Spearman-Rho, Wilcoxon Signed-Rank, Mann-Whitney-Wilcoxon . . . 36
4.10 Overview . . . 42

5 Conclusion **45**

6 Future work **47**

Bibliography **49**

1 Introduction

Covert channels are a form of hidden communication, that alters certain values not intended for communication, to establish transmission of information. In a computer network scenario the network traffic is altered to create this form of communication. One of the two main methods to hide communication is to hide the information within the network packets, such as adding information to the timestamp field. The other method is to use the time between subsequent packets, the inter-packet delay (IPD), to transfer information. The latter form of covert channels are called covert timing channels (CTCs), and they are what this thesis will focus on. The method of modulating the time between packets can be very simple, by having set delays, up to basing the delays on models of legitimate traffic, in order to mask this communication taking place. The CTCs can be used in any place where there is network traffic present, with a wide variety of possible applications. One such an example is the use of covert channels in data exfiltration, from within a company, where the use of covert channels would allow this to happen undetected. Since CTCs can be created anywhere that network traffic exists, there is a risk that data leakage or other unwanted communication can occur. It is thus important to be able to detect the presence of covert channels, in order to disrupt this communication taking place.

Previous work on the subject of CTC detection has introduced a significant amount of detection methods, with high reported performance [2, 8, 16]. However, each work mainly focuses on introducing new detection methods and only uses some previously presented methods to compare these to their performance. In all of the research each detection method is only performed on a handful of different CTCs, with a limited amount of variation in the used settings. There is also little reproduction of the results of previous work in realistic network scenarios, so there is no significant confirmation for the reported performances. Due to these points there is no clear overview of the overall performance of detection methods for all covert channel techniques. Next to this, there has been little research done on the impact of network jitter on the performance of detection methods. The main form we have seen in which jitter is added to the traffic is by replaying it over a single real or simulated network. This means that there is no comparison between different amounts of jitter, and how the detection methods are affected by different intensities of this network effect. Further, the data that is used to represent the legitimate traffic is largely comprised of traffic taken from certain models based on statistical distributions, that are assumed to represent real traffic. When real traffic is used in the performance analyses, this is mostly either a small amount or it is taken from datasets that might not represent current traffic. We would like to provide insight into the current state of detection methods for covert timing channels, and their applicability in real network scenarios. For this purpose we have formulated the following research questions, from the above identified gaps in current research for this subject:

Research question 1: Is there a single detection method or combination of detection methods that can sufficiently detect the most prevalent covert timing channels?

Research question 2: How does the existence of network effects, such as jitter, affect the performance of detection methods?

To provide answers to these research questions a broad evaluation of the current detection methods is performed, under a common methodology. With the evaluation we attempt to reproduce the results obtained in previous work, and examine the strengths and weaknesses of the detection methods. To perform the tests, covert packets are injected in recorded HTTPS and SSH traffic from the TU Delft, with varying network conditions, in the form of simulated network jitter. The detection methods are performed on this traffic, and results are obtained in the form of true, and false positive rates for each combination of detection method (18) and covert channel (8). This thus resulted in 144 analyses for both examined traffic types, with multiple variants for each covert channel technique. From the results we can make conclusions on the current state of detection methods, for the detection of a broad range of CTCs, using these (combinations of) metrics. Next to this, the use of real network traffic and the inclusion of jitter shows the performance of the detection methods, as it would when applied in a more realistic network scenario.

The remainder of the thesis is structured as follows: Section 2 gives an overview of the prevalent CTC techniques and detection methods, and provides a summary of what is currently covered. Next, Section 3 explains how tests are performed to obtain results, as well as provide more detail on the used data, and settings for the CTCs and detection methods. Following from the performed tests, Section 4 gives an in-depth analysis into the performance of the detection methods on the various CTCs. Section 5 provides answers to the aforementioned research questions on the current state of detection methods. Finally, in Section 6 the possible directions of further research on this subject are identified and discussed.

2

Background and related work

A covert channel, also called network steganography or information hiding, is a communication method that is achieved by using channels that are not intended for information transfer. Compared to a method such as cryptography, which tries to make the message unreadable for everyone that is not supposed to receive the message, a covert channel hides the fact that communication is taking place altogether, by having the communication mixed in with with the normal operation of a process. The first definition of a covert channel in this sense is given in 1973 by Lampson [10], where they discuss possible data leakage that can result from the existence of these channels. Lampson notes that even though a confined program might be secure against unauthorized access, there are still subtle ways in which this program may leak data. A program might, for example, encode the information in the bill the program generates for the use of the service, or place or remove a read/write lock on a file in certain time intervals. From this action another program could then see whether or not there is a lock on the file, which would indicate a transmitted boolean value. In 1984 Simmons describes a model for covert channels in a malicious scenario [19]. In this scenario two prisoners intend to escape the prison, and are attempting to communicate their plan. To communicate they are able to relay messages to each other through intermediate parties, but these messages can be read and altered by the prison warden. If their plan is found out, the warden will lock them up in solitary confinement and they will not be able to escape. The prisoners therefore have to hide the information for their escape in the messages, in such a way that it is difficult for the warden to detect, which is the covert channel in this scenario.

Covert channels can be broadly classified into covert storage channels, which the aforementioned scenarios fall into, and covert timing channels. In the context of computer networks, storage channels use direct or indirect writing of object values in network packets, such as the timestamp or unused header values, to transmit data. A timing channel on the other hand manipulates the timing or ordering of network events as a communication channel. In both cases an attacker can be situated inbetween unsuspecting users of an overt channel, and manipulate their traffic, or generate and change its own traffic to create a covert channel. It is also possible that the attacker is both the sender and the receiver of the channel, such as extracting information from a different location to their own computer. Covert channels can be used in a variety of scenarios where communication is desired, and attackers are very creative in finding ways to implement them in places where this communication is not intended. It should be noted however that a covert channel is not inherently malicious, much like how the use of cryptography is not inherently malicious. An application is presented by Nagaraja et al. [13], where they show a covert channel using images with encoded information, that are uploaded to social media. Infected users that are connected to each other on the social media platform can extract the information from the uploaded images and communicate covertly through them. This communication is then used for the purpose of command-and-control messages for the botnet of infected users. Another example is data exfiltration from a high security environment, which has access to privileged information, to a low security environment. In this case low can send information to high, but not the other way around, with the exception of acknowledgement responses for the receipt of low's messages. This restriction on the communication should prevent high from leaking data to low. However, a covert channel can be created by modulating the timing of the acknowledgements and in this way high can communicate the privileged information to low. The main downside of covert channels is that the amount of information

that can be transmitted per second is very limited, with channel capacity in the order of bits per second. If the covert channel makes use of overt traffic from an unsuspecting user, this limits the capacity even further. On the other hand, if the channel can stay undetected it can slowly transfer information over a longer period of time, which might be more lucrative than other means.

Hindering malicious covert channels mainly looks at the detection, elimination, and capacity limitation of these channels. By detecting the use of covert channels (early), infected hosts can be found and stopped, which will limit the amount of data that can be transferred. Eliminating covert channels involves finding security flaws that can be exploited, and removing these opportunities for creating covert channels. For instance in the case of unused bits in packet headers, there can be a default value ascribed to them so that they cannot be used for covert channel purposes anymore. Next to this, due to the limited channel capacity of covert channels, time-sensitive data is hard to acquire before it is not relevant anymore. If the capacity can be limited even more, then this data cannot be extracted in time. Since there are such a large number of possible covert channels this thesis focuses on covert timing channels in computer networks, meaning those that use the time between network packets for information transfer. The rest of this chapter discusses prevalent covert timing channels and the related work on evaluation of detection methods for these channels.

2.1. Covert timing channels

2.1.1. Simple Covert Channel

One of the first and simplest covert timing channels (CTCs) is the aptly named Simple Covert Channel (SCC) [11]. The SCC consists of a set of symbols $\{s_1, ..., s_k\}$ and the related output alphabet $\{t_1, ..., t_k\}$ of delay times, with $t_j < t_{j'}$ if $j < j'$. This creates a k-symbol channel, with k different inter-packet delays (IPD), which is the time between subsequent packets. The sender and receiver agree on a time or the occurrence of a network event, such as the first packet being sent. When this time is reached or the event has occurred, the sender starts transmitting information to the receiver. The sender does this by selecting the corresponding delay from the output alphabet for each symbol that needs to be sent, and transmits a packet for each symbol such that the time between the previous packet and the following packet sent is equal to the selected IPD. The receiver can then compare the perceived IPD of the received packet to the output alphabet and look up the transmitted symbol. Due to various delays in the network it is unlikely that the receiver will record an IPD with the exact amount as it was sent, so the receiver chooses the IPD that is closest in the output alphabet. A binary form of the SCC would use two symbols, $s_1 = 0$ and $s_2 = 1$, and two static delays, t_1 and t_2. When a zero is to be transmitted the delay between the subsequent packets will be t_1, and t_2 for a one. Figure 2.1 shows an example of such a binary SCC, which transmits the string ”010100”.

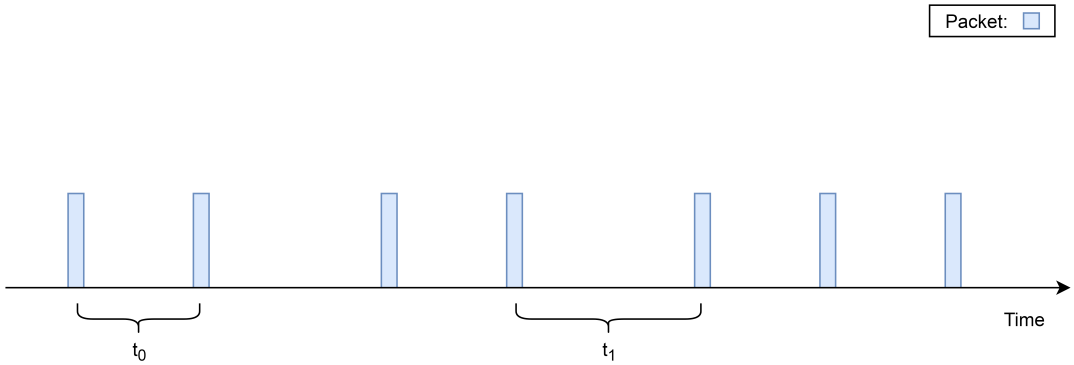


Figure 2.1: Simple covert channel example

2.1.2. On-Off Covert Timing Channel

The On-Off CTC [5] is a solely binary covert channel that takes a different approach from the SCC, where it uses a time slot to send its covert traffic. For each time slot either no packets or a single packet is sent over the network. The size of the time slot can be agreed upon beforehand by the sender and

receiver, or a default size can be used to exchange a new time slot at the start of the channel. To communicate a zero the CTC stays silent during the entire time slot, and sends a single packet in the middle of the time slot to transmit a one. The receiver can then receive covert communication by listening for packets during each time slot. If there are no packets during the time slot it records a zero and waits for the next time slot to end. However if there is a packet during the time slot it records a one and shifts the start and end of the next time slot earlier or later, under the assumption that the packet occurred exactly in the middle of the last time slot. This will help protect the channel against desynchronization, for example due to jitter in the network, and aligns the timing windows of the sender and receiver. Figure 2.2 shows an example of an On-Off covert timing channel, with indicated time slots, which transmits the string ”10110101”.

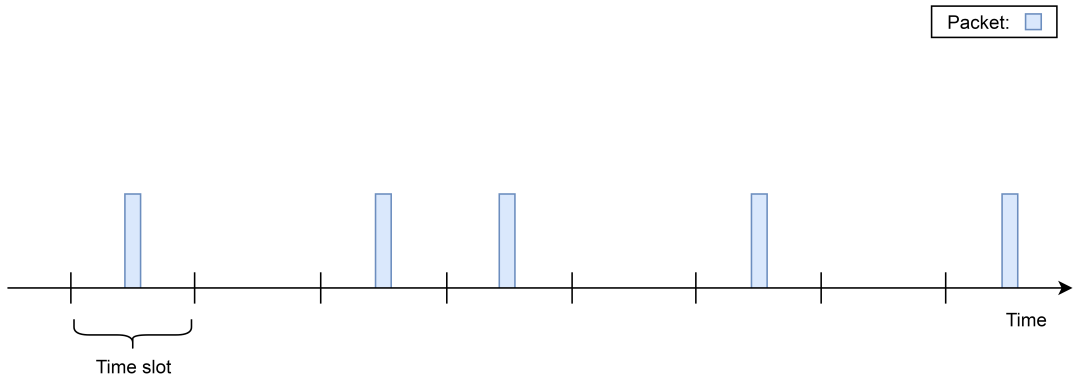
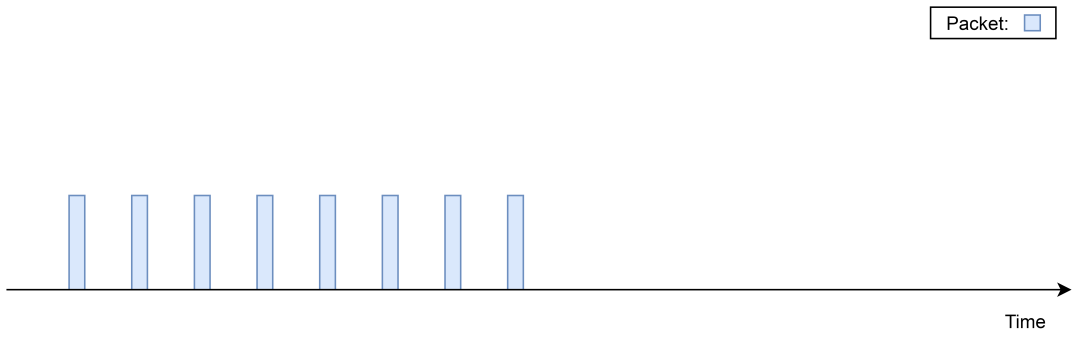


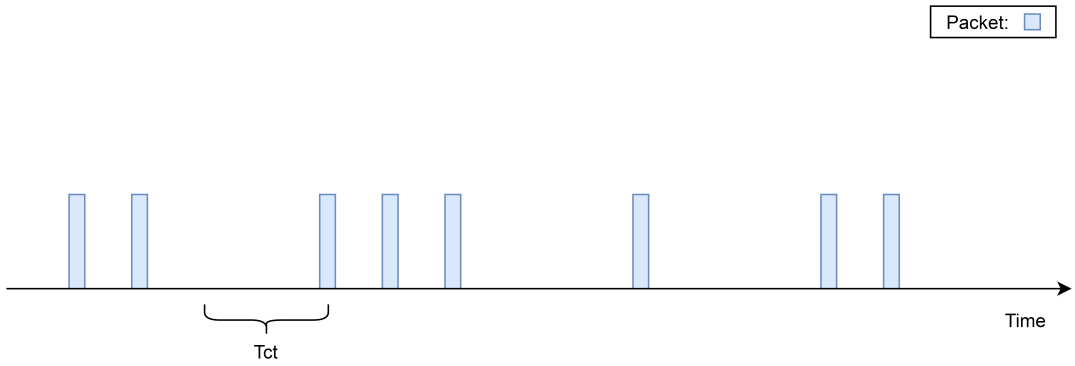
Figure 2.2: On-off timing channel example

2.1.3. Delayed Packet One Indicator

Delayed Packet One Indicator (DPOI) [17] is a binary covert channel that aims to improve on the On-Off CTC, wherein it does not utilize the silence periods that are present in the On-Off CTC. By not using silence intervals DPOI can send more packets, and thus transmit more covert information, in the same intervals as On-Off CTC, and therefore DPOI can achieve improvements such as a higher channel capacity. The covert channel requires a (legitimate) overt traffic stream, which can be created by the covert sender themselves, and a mutual covert timing value T_{ct} between the sender and receiver. For this CTC technique the use case is more aimed towards a man-in-the-middle approach, where the process of the covert channel alters the network traffic of another user. While others can also be used in this way, it is much more difficult to obtain the exact values required for them, from adding delays to existing traffic. Using this stream of traffic and their IPDs, each packet is sent without a delay to transmit a zero, and the delay T_{ct} is added to IPD of the subsequent packet to signal a one. In this case the first packet sent does not transmit any covert information, but acts as a starting packet to acquire the first IPD between the first and second packet. Receiving is done by running a timer for the time T_{ct} and waiting to receive a packet. If a packet is received before the timer runs out, a zero is recorded and the timer is reset for receiving the next packet. When a packet is received after the timer has run out, a one is recorded and the timer is reset as well. In Figure 2.3 an example is shown of a DPOI covert channel transmitting the string ”0100110”, with 2.3a the original traffic with equal IPDs, and 2.3b the covert channel traffic with added delay T_{ct}.



(a) The original traffic



(b) The delayed DPOI traffic

Figure 2.3: Delayed Packet One Indicator example

2.1.4. JitterBug

The JitterBug (JB) [18] covert channel was designed to intercept interactive network application traffic, such as keystroke input in SSH, as added malicious hardware to a keyboard. It should however be noted that the covert channel can also be implemented in software and thus such a device is not necessary, and the intercepted traffic can be replaced by pre-recorded or self-generated traffic. JB transmits data covertly by applying a delay to the intercepted traffic before sending it over the network. For covert communication an integer timing window w and a random number generator seed are known between the sender and the receiver. To encode a binary sequence $\{b_i\}$ the intercepted sequence $\{t_i\}$ of times when an event occurs are each delayed by a time τ_i. The new sequence then becomes $\{t_i^{'}\}$, with $t_i^{'} = t_i + \tau_i$ and $\tau_i < w$. The IPDs δ_i, with $\delta_i = t_i^{'} - t_{i-1}^{'}$, that are applied to the packets are calculated such that:

$$\delta_i \bmod w = \begin{cases} 0 & \text{if } b_i = 0; \\ \lfloor w/2 \rfloor & \text{if } b_i = 1; \end{cases}$$

The intercepted packets are thus delayed such that the IPD in milliseconds modulo w is equal to 0 to transmit a zero, or the IPD modulo w is equal to $\lfloor w/2 \rfloor$ to transmit a one. Additionally, to avoid regularity in the time between packets, a random delay s_i (with $0 \leq s_i < w$) is added to the IPD, which is generated using the seed value. For example, given a sequence of intercepted IPDs in milliseconds $\{88, 164, 204, 73, 120\}$ and a timing window $w = 20$, a sequence of bits $\{0, 1, 0, 1, 0\}$ would be given by the IPDs $\{100, 170, 220, 90, 120\}$. These values are then each adjusted with the added random delay, which removes the perceived regularity. The receiver in turn records the perceived times of the packets $\{\widehat{t_i}\}$ and can obtain the sequence of IPDs $\{\widehat{\delta_i} = \widehat{t_i} - \widehat{t_{i-1}}\}$. If the added random delay is used, the receiver

can generate this value with the same random seed and simply subtract it from the received IPD. From the sequence of IPDs the received symbols are decoded as follows:

$$\text{if} \quad -\varepsilon < \widehat{\delta_i} \leq \varepsilon \quad (\text{mod } w) \text{ then } b_i = 0;$$
$$\text{if } w/2 - \varepsilon < \widehat{\delta_i} \leq w/2 + \varepsilon \ (\text{mod } w) \text{ then } b_i = 1;$$

The value ε is the parameter that sets the guard bands for compensating against variations in perceived delays, for example due to network jitter. This decides whether a received IPD is considered a zero or a one, when it is not strictly the value that is expected. It can be adjusted to accommodate different network conditions. Note that the value of ε cannot exceed $w/4$, or else the windows of receiving either zero or one would overlap.

2.1.5. Model Based Covert Timing Channel

Model Based-CTC (MB-CTC) [9] models the IPDs of sent packets on a probability distribution created from observed legitimate traffic. By modeling the covert channel traffic on legitimate network packets, it can closely resemble this traffic and is more likely to avoid detection. MB-CTC first monitors the background network traffic and filters for the specific type of traffic that is to be mimicked. For example, by filtering for HTTP traffic, when such traffic will be used for the covert traffic, it will give a closer model for that type of traffic. Next, the filtered traffic is fitted in sets of 100 IPDs to the Exponential, Gamma, Pareto, Lognormal, Poisson, and Weibull distributions, using maximum likelihood estimation. More models can be added to fit different traffic and network conditions as needed, if such a model more accurately describes the traffic. The accuracy of the models is compared by the root mean squared error (RMSE) of the models compared to the IPDs, and the model with the lowest error is chosen as the used model. The RMSE of a model and a set of IPDs can be acquired by taking samples from the model (with equal size to the IPDs) and calculating the RMSE of the samples and the IPDs. To account for randomness of sampling and the variation in RMSE values that may cause, it may be required to sample multiple times and take the mean of the calculated RMSEs. Since traffic characteristics can change over time, MB-CTC also has the possibility to modify its model after sending 100 covert IPDs, with new packets filtered from live traffic and again fitted to the distributions. When there is a change in the model these new parameters are sent to the receiver. The aforementioned filtering and fitting steps can be performed in two ways, namely in an online or offline mode. In the case of the offline mode, both sender and receiver are in possession of an identical, pre-recorded traffic sample. Then from this sample they obtain the same model required for communication. For the online mode the sender acquires a model from live traffic, which is then communicated to the receiver. The offline mode uses less resources, but the retrieved model may not accurately represent the current network behaviour. On the other hand, the online mode requires a startup protocol to transmit the model from the live traffic to the receiver. After filtering and fitting the model, the information that is to be sent is encoded using the parameters from the model. The discrete symbol s that needs to be transmitted is first continuized into a value between 0 and 1. By using a seeded uniform random number r (with $0 \leq r < 1$) for each symbol, and the amount of possible symbols |S|, the continuized symbol r_s is calculated by:

$$F_{continuize}(s) = (s/|S| + r) \bmod 1 = r_s$$

From the continuized symbol the delay d_s is obtained, which is used as the IPD for the subsequent packet. Note that a starting packet needs to be used to compare the time between packets and calculate the first IPD. With the inverse distribution function of the model and its parameters, the encoded delay is obtained:

$$F_{encode} = F^{-1}_{model}(r_s) = d_s$$

To retrieve the sent symbol from the received delays the receiver inverts the operations done by the sender. First the received delay d_s is decoded by using the inverse of the inverse distribution function, which is the cumulative distribution function of the model. The continuized symbol r_s is then calculated by:

$$F_{decode} = F_{model}(d_s) = r_s$$

The transmitted symbol s is then obtained by reverting the continuized symbol r_s into a discrete symbol, using the same seeded random number r:

$$F_{discretize}(r_s) = |S| \cdot ((r_s - r) \bmod 1) = s$$

2.1.6. Repeat Model Based Covert Timing Channel

The Repeat Model Based-CTC (RMB-CTC) [12] is a slight variation on MB-CTC, where the covert channel uses added repeat bits to attempt to evade entropy-based detection measures. After each two packets that are sent, one of the two will be randomly selected with the same probability to be repeated twice more using the same IPD. So for each two subsequent IPDs $\{d_1, d_2\}$, each sequence of four IPDs will be either $\{d_1, d_2, d_1, d_1\}$ or $\{d_1, d_2, d_2, d_2\}$. Transmitting and receiving symbols works in the same way as MB-CTC, except that the receiver can ignore the IPD of the last two packets in the four packet sequence. The downside of using repeat bits is that it decreases the channel capacity of the covert channel, because the two repeated packets carry no additional information. It was found by Gianvecchio et al. [8] that MB-CTC has a higher entropy value than legitimate traffic, which has some repeating patterns reducing this value. This means that MB-CTC can successfully be detected by entropy detection metrics. Since entropy is a measurement of complexity, adding redundant information lowers the complexity of the measured process and can thus more likely avoid detection by entropy-based detection methods.

2.1.7. Time Replay Covert Timing Channel

Time Replay-CTC (TR-CTC) [4] takes a different approach to mimicking legitimate network traffic, by using a set of recorded legitimate packets. From the recorded traffic the IPDs for each flow are sorted into bins according to a rule-set and both the sender and receiver have access to the same set of recorded traffic and rule-sets. TR-CTC has three different types of rule matching, namely binary-matching, rule-matching, and exact-matching channels. The binary-matching channel only uses a single rule in the form of $< 0, \tau_{cutoff}, zero >$, which indicates that IPDs that are smaller than τ_{cutoff} are sorted into the bin for the symbol zero and all other IPDs are sorted into the bin for the symbol one. To communicate information covertly, the sender first transmits a starting packet. Then for each symbol it randomly draws (and removes) a delay from the corresponding bin, which is applied as the IPD for the subsequent packet. The receiver can then calculate the received IPD $\tau_{observed}$ and matches it with the rule. If $\tau_{observed} < \tau_{cutoff}$ then the receiver records a zero and a one otherwise. Alternatively a buffered cutoff value can be used to increase channel accuracy. In this case the receiver records a zero if $\tau_{observed} \leq \tau_{cutoff} - \delta$, a one if $\tau_{observed} \geq \tau_{cutoff} + \delta$, and nothing otherwise, with δ being the chosen buffer value. Rule-matching in turn can have two or more rules in the form of $< \tau_1, \tau_2, one >$, such that an IPD d is sorted into the bin for symbol one if the rule matches $\tau_1 \leq d < \tau_2$. Like the binary-matching channel a symbol is transmitted by randomly drawing a delay from the related bin. The receiver matches the received IPD $\tau_{observed}$ to every rule and records the symbol indicated by the rule that matches $\tau_1 \leq \tau_{observed} < \tau_2$. The exact-matching variant has a rule for every possible IPD (i.e. $< \tau_1, \tau_1, one >$), for which the symbols are randomly assigned. The recorded traffic is then sorted into the bin for the symbol where the delay $d = \tau_1$. Transmitting a symbol works the same as the previous channel types, and the receiver matches the $\tau_{observed}$ exactly to the rule where $\tau_{observed} = \tau_1$. Where applicable the thresholds of rules can also be the median or maximum values of the IPDs (e.g. $< mean, max, zero >$), which both the sender and receiver can calculate from the recorded traffic.

2.1.8. L-Bits-to-N-Packets

The L-Bits-to-N-Packets (LBtNP) [4] covert channel, as the name suggests, encodes a sequence of L covert bits as N subsequently transmitted IPDs. The sender and receiver both have knowledge of the amount of bits L, the number of packets N, the minimum delay Δ, and the incremental delay δ. For each 2^L combinations of L bits a unique sequence of N IPDs $(T_1, T_2, ..., T_N)$ is recorded in the code book for the sender and receiver. These times are selected from the set $E = \{T : \Delta + k * \delta, k = 0, 1, ...\}$, where the maximum of k has to be sufficiently large so that each bit combination can have a unique sequence (i.e. $(k+1)^N \geq 2^L$). The covert communication starts with a starting packet, which carries no information.

Then when a sequence of L bits is to be transmitted, the N packet delays are looked up in the code book and sent subsequently as IPDs for the packets. From the observed packet timings the receiver can construct sequences of N IPDs. Then by matching a sequence to the closest mapping in the code book the receiver can recover the L bits that were transmitted. To avoid regularity in the IPDs of the packets it is also possible to add a seeded random value to each IPD, in which case the sender and receiver also need to possess the same seed. Before sending an IPD a random value v, uniform in $(0, \delta)$, is generated and added to the delay. The receiver can then simply generate the same random value with the seed and subtract it from the perceived delay, after which the decoding operations continue as normal. A variation of the standard scheme of the covert channel is also proposed, in which the sequence of IPDs for each L bits is modeled to a distribution. The distribution can be chosen such that it models certain types of traffic that the covert channel is trying to mimic. Instead of directly mapping permutations of bits to timings they are mapped to sequences of length N $(\frac{k_1}{K}, \frac{k_2}{K}, ..., \frac{k_N}{K})$, with $0 \leq k_i \leq K-1$ and K chosen such that $K^N >= 2^L$. When transmitting L bits the sequence is looked up in the code book as $(x_1, ..., x_N)$. Using a seeded random number generator each x_i in the sequence is masked by a value u_i, uniform between 0 and 1. The masking for each entry is calculated as $r_i = x_i \oplus u_i \triangleq x_i + u_i \bmod 1$, resulting in the masked entries $(r_1, ..., r_N)$. The final sequence of IPDs $(T_1, ..., T_N)$ is obtained by calculating $T_i = F^{-1}(r_i)$ for each masked value, using the inverse distribution function of the chosen model. This sequence is then used as the IPDs for the N sent packets. From the observed packet timings the receiver can again construct sequences of N IPDs $(R_1, ..., R_N)$, after which the operations performed by the sender are reversed. First from the same generated seeded random values u_i and the cumulative distribution function of the model F, the value $x_i^* = F(R_i) \oplus (1 - u_i)$ is calculated for each IPD. These values are then rounded to the nearest $\frac{k}{K}$, computed as $x_i^d = \lfloor K \cdot x_i^* + 0.5 \rfloor / K$. And lastly the sequence $(x_1^d, ..., x_N^d)$ is looked up in the code book, which returns the sent L bits.

2.2. Detection methods

2.2.1. ε-Similarity, Regularity, and Compressibility

One of the first scientific research on detection methods for covert timing channels is given by Cabuk et al. [5]. In their paper they evaluate the efficacy of two metrics for the detection of the On-Off covert timing channel. They also vary the timings of the covert channel during operation and add noise to the channel, to compare how well the metrics work under those scenarios. The two detection methods under evaluation are regularity and ε-similarity. The regularity metric examines whether the variance in the IPDs remains constant over a certain window. To calculate this, the traffic is separated into non-overlapping windows of size w, for each window i the standard deviation σ_i is computed, and for each pair of windows i<j the pairwise relative difference of σ_i and σ_j is calculated. The regularity metric is then the standard deviation of these pairwise differences. Formally this is: $regularity = STDDEV(\frac{|\sigma_i - \sigma_j|}{\sigma_i}, i < j, \forall i, j)$. The other measure is ε-similarity, which is the percentage of relative differences that are less than ε. For a sample of packets the IPD for every packet is calculated, given by P_i, and the IPDs are sorted in increasing order. Then from the sorted IPDs the relative difference, $|P_i - P_{i+1}|/P_i$, for each pair of consecutive points can be calculated. The percentage of relative differences that is smaller than ε is then the ε-similarity. The values for ε and the thresholds for ε-similarity and regularity can be determined from legitimate traffic, the comparison to these thresholds will then indicate whether a sample is deemed a covert channel or not. Cabuk et al. report regularity scores on the standard On-Off channel with 100% true positive rates and 0% false positive, but very low scores on channels with varying timings or channels with added noise. They also report high ε-similarity detection scores for most categories, except where there are high levels of added noise. In a subsequent paper they also propose the compressibility metric as an extension to their previous work [6]. The compressibility metric is an approximation of the Kolmogorov complexity of a string S, which is the shortest universal computer program that produces this string. The metric is computed as $\kappa(S) = \frac{|S|}{|C|}$, where $|S|$ is the length of the original string S derived from a sample of IPDs and $|C|$ is the length of the compressed string of S. This value is then compared to a threshold obtained from the compressibility values from legitimate traffic. The metric is evaluated on the same variations of On-Off channel. For compressibility Cabuk et al. report similarly high detection scores as with ε-similarity, but note a better performance on higher noise levels.

2.2.2. Arimoto-Blahut, and Mean IPD

In their paper Berk et al. give two detection methods, one based on the Arimoto-Blahut algorithm and another using the mean inter-packet delays [3]. These methods are evaluated against binary and multi-symbol simple covert channels. The mean IPD method is based on the premise that (for a binary channel) the counted IPDs will center around two distinct values. For the covert channel the sample mean of the IPDs will lie somewhere between two spikes (the two binary channel delays), but this can also be extended to channels that use symbols in multiples of two. For legitimate traffic the mean of the IPDs will be in the center of the large spike. The metric is calculated using: $P_{CovChan} = 1 - \frac{C_\mu}{C_{max}}$, where C_μ is the packet count at the mean, and C_{max} is the largest packet count of the histogram. The Arimoto-Blahut requires a given a channel matrix, which contains the probabilities of receiving delay j when delay i is sent, for a certain range of delays. Using this channel matrix the algorithm can then estimate the optimal symbol distribution, to achieve the highest channel capacity. The traffic under analysis is compared to the optimal symbol distribution from the algorithm. If it (closely) matches the distribution, then it is considered a covert channel. This distinction is made under the assumption that an attacker will choose their input such that the channel capacity is maximized. Berk et al. report that the methods perform well, but unfortunately do not include any detection scores in their work.

2.2.3. Kolmogorov-Smirnov, Entropy, and Corrected Conditional Entropy

On the subject of network packet watermarking, which can be considered a form of covert timing channel communication, Peng et al. apply the Kolmogorov-Smirnov (KS) test as a detection method for watermarked IPDs [14]. The KS-test determines whether or not two samples (or a sample and a distribution) differ. The statistic measures the maximum distance between two empirical distribution functions, which is given by: $KSTEST = max|S1(x) - S2(x)|$. The KS test is applied by comparing training data from legitimate traffic against possible covert traffic and determining if the distributions are close or not. Gianvecchio et al. further evaluate the KS test and regularity metric on the On-Off, Time Replay, and JitterBug covert channels, and propose the first order entropy (EN) and corrected conditional entropy (CCE) detection metrics [7]. In an extension to this paper Gianvecchio et al. also evaluate the aforementioned detection methods on the MB-CTC [8]. The proposed entropy metrics aim to measure the regularity or complexity of a process. The first order entropy measures the entropy of single IPDs, meaning patterns of length one. EN is given as

$$H(X_1, ..., X_m) = -\Sigma_{X_1,...,X_m} P(X_1, ..., X_m) log_2 P(X_1, ..., X_m)$$

where $P(X_1, ..., X_m)$ is the conditional probability $P(X_1 = x_1, ..., X_m = x_m)$. The corrected conditional entropy (CCE) estimates the entropy rate of patterns of length m in the IPD sample. The CCE is given by:

$$CCE(X_m|X_{m-1}) = CE(X_m|X_{m-1}) + perc(X_m) \cdot EN(X_1)$$

where CE is the conditional entropy, $perc(X_m)$ is the percentage of unique patterns of length m, and $EN(X_1)$ is the entropy of the sample with pattern length one. The aim is to find the minimum value of CCE, by increasing m until it is found, which is the closest estimate of the entropy rate. EN and CCE values from possible covert traffic can then be compared to thresholds obtained from legitimate data, to indicate the presence of a covert channel. The true positive detection results reported by Gianvecchio et al. for each covert channel are obtained by setting the false positive rates to 0.01 during training. The thresholds are then adjusted, so that legitimate traffic is not labeled as a covert channel higher than this rate. For the On-Off channel all detection methods except regularity reach 100% true positive detection rates. On the other hand for the TR-CTC and MB-CTC only CCE performs well, with near full true positive rates. Next to this EN is the only one that has 100% true positive rates for JitterBug detection, with the rest of the methods lower than 5%.

2.2.4. Support Vector Machine

Mou et al. present a method in which they apply Support Vector Machines (SVM) to the field of covert channel detection [12]. An SVM is a pattern recognition technique in the form of a supervised learning model, which can take multiple detection metrics to learn and classify covert traffic from legitimate traffic. In this case a wavelet transform is used to get a decomposition of IPDs into different levels. From this decomposition the maximum entropy at every level and the percentage of energy of the first level is retrieved. These values are then used for classification in the SVM, which can use them to

make a comparison between legitimate and covert traffic. The detection of the SVM is performed on SCC, TR-CTC, MB-CTC, and RMB-CTC with 100% detection accuracy for all except MB-CTC, which still has a high precision of 96.5%. Next to this Mou et al. also show that the SVM can distinguish the different covert channels that are being used, with high confidence. Further to verify that their variation on model-based variant RMB-CTC is more likely to avoid entropy-based measures, the EN and CCE methods are also used for detection of RMB-CTC. These metrics give detection accuracy of 79% and 64.75% respectively, which is relatively low compare to the SVM detection method. On the other hand the main downsides of using SVMs is that they require extensive calculations and knowledge of other detection metrics for the use in the classifier.

2.2.5. Kullback-Leibler divergence, Skew, Kurtosis, and Welch's t-test

In their work Archibald et al. discuss the addition of fountain codes for MB-CTC communication, for the purpose of robustness against network jitter and disruptions [1]. To quantify the security against detection for this method, they employ the KS test and propose the Kullback-Leibler divergence (KLD) as a metric. The KLD is the relative entropy between two (samples of) probability distributions, P and G. KLD defines the minimum additional information, measured in logarithmic scale, needed for one distribution to perfectly model another, providing a measure of how much these distributions differ. This metric can be used to compare a distribution of legitimate traffic against a distribution of possible covert traffic, to be able to differentiate between these different types of traffic. The KLD measure is given by: $D_{KL}(P||G) = \sum_x p(x) \cdot log(\frac{p(x)}{g(x)})$. In a later study Archibald et al. also perform an extensive evaluation of detection methods on the covert timing channels JB, MB-CTC, and TR-CTC [2]. They apply the KS test, regularity, EN, CCE, and KLD metrics from previous works on these different covert channels and measure their efficacy. Next to this they also propose the Welch's t-test, skew, and kurtosis metrics, as well as SVM detection using these skew and kurtosis metrics for MB-CTC. Since JB creates its covert communication by shifting the IPD distribution of application traffic, metrics that cannot detect these shifts are not suited for detection of this covert channel. Among these metrics are EN, KLD, kurtosis, and skew, which are therefore not applied for the detection of JB traffic. The proposed Welch's t-test tests the null-hypothesis of whether two population means are equal. A sample from legitimate traffic is compared against possible covert traffic, and the p-value then indicates whether this null-hypothesis can be rejected. The main benefit is that the test does not require equal sample sizes nor does it expect equal variance of the samples. This means that a larger sample of training data can be used for a better fingerprint of legitimate traffic. The t-value of the Welch's t-test is given by:

$$t = \frac{\overline{X_1} - \overline{X_2}}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}}$$

Further the kurtosis of a distribution describes the extremity of the values around the tail ends of the distribution, and is given by $Kurt[X] = E[(\frac{X-\mu}{\sigma})^4]$. The skew of a distribution measures where the area of the distribution falls, to the left or right of the mean, and is given by $Skew[X] = E[(\frac{X-\mu}{\sigma})^3]$. From a set of IPDs a kurtosis or skew vector can be created and from this vector the mean, standard deviation, and regularity is extracted. Archibald et al. report high detection accuracy for the Welch's t-test on JB and MB-CTC, as well as high accuracy for CCE on TR-CTC and MB-CTC, both ranging in precision from 86% to 97%. The SVM for the detection of MB-CTC traffic showed increased accuracy with more detection metrics used for classification, to a score of around 90%. Results obtained from the other detection methods scored relatively low on average compared to these methods.

2.2.6. Spearman-Rho, Wilcoxon Signed-Rank, and Mann-Whitney-Wilcoxon

One of the more recent studies is done by Rezaei et al. where they propose the use of non-parametric tests for covert channel detection [15]. The tests they bring forth are Spearman-Rho (SRHO) and Wilcoxon Signed-Rank (WSR) tests, which are used to detect On-Off, LBtNP, JB, and TR-CTC. In addition they also published an extension to this paper, which includes a third non-parametric tests, the Mann-Whitney-Wilcoxon (MWW) Rank Sum test, and added DPOI as a covert channel technique under evaluation [16]. The SRHO, WSR, and MWW tests are non-parametric statistical tests which use the rank of observed samples of IPDs, instead of their values. These ranks are then used to

determine if two samples are drawn from the same distribution. The implementation using these tests takes samples of two subsequent blocks of IPDs from the same flow, for which the p-values of these tests are calculated. Both blocks are then updated by a given step size, where part of the second block is added to the first, and new IPDs are added to the second block. Calculating the p-values and updating the sets is repeated multiple times, to observe changes in the traffic. The average of these p-values for each test is compared against a threshold, formed from performing the tests on flows of legitimate traffic, which determines whether the traffic is classified as covert. The main idea is that for flows of covert traffic the p-values will likely be large, compared to those of legitimate traffic. This is due to the fact that the created covert traffic will have similar IPDs over the entire flow, whether in value or distribution, which results in higher overall p-values. In the case of legitimate traffic, there can be more differences over time within each flow, which in turn leads to lower p-values. The evaluation is performed over 3 different settings for each covert channel technique and 8 settings for each detection metric. The average true positives and false negatives for the reported values are 96.4% and 7.0% for WSR, 89.0% and 8.4% for SRHO, and 99.4% and 3.9% for MWW.

2.2.7. Summary

The detection techniques presented in these previous works show promise for the detection of CTCs, with high levels of accuracy over different channels. There are however some drawbacks, where some methods require a lot of computational power or are only able to detect some covert channels reliably. In a realistic scenario the type of covert channel the attacker will use is likely unknown. If a detection method is used which does not detect that type very well, then the covert traffic might go by unnoticed. It is therefore beneficial to recognize which metrics, or combinations of metric, do sufficiently well in general detection over all variations of CTCs, instead of high accuracy in singular CTCs. Table 2.1 gives an overview of the related work and shows which detection methods have been applied to which CTC. From this table it can be seen where there are still gaps missing in the evaluation of the detection methods.

	SCC	On-Off	DPOI	JB	MB-CTC	RMB-CTC	TR-CTC	LBtNP
Regularity		[5, 7, 8]		[2, 7, 8]	[2, 8]		[2, 7, 8]	
ε-similarity		[5, 6]						
Compressibility		[6]						
Arimoto-Blahut	[3]							
Mean IPD	[3]							
KS test		[7, 8]		[2, 7, 8]	[2, 8]		[2, 7, 8]	
EN		[7, 8]		[7, 8]	[2, 8]	[12]	[2, 7, 8]	
CCE		[7, 8]		[2, 7, 8]	[2, 8]	[12]	[2, 7, 8]	
SVM	[12]				[2, 12]	[12]	[12]	
KLD					[2]		[2]	
Welch's t-test				[2]	[2]		[2]	
Kurtosis					[2]		[2]	
Skew					[2]		[2]	
SRHO		[15, 16]	[16]	[15, 16]			[15, 16]	[15, 16]
WSR		[15, 16]	[16]	[15, 16]			[15, 16]	[15, 16]
MWW		[16]	[16]	[16]			[16]	[16]

Table 2.1: Detection method evaluation overview

3

Experimental evaluation

In order to perform a broad evaluation on the performance of the aforementioned detection methods, we apply the detection methods to every covert channel technique in Section 2.1, in varying network conditions. Simulated network jitter is applied to the traffic, to replicate the varying conditions present in different networks. From performing the detection methods on these scenarios of network traffic and CTCs, we can quantify their performance and examine the results. This section will expand on how the data, that is used for the evaluation, is obtained and processed, as well as how the evaluation is performed and how the results are obtained from them. Next to this, an overview is given for the parameters and settings for the covert channels and detection methods.

3.1. Dataset

The data used for the evaluation is network traffic recorded on the TU Delft network, from the building of the faculty of Electrical Engineering, Mathematics, and Computer Science. All the HTTPS and SSH traffic of the building was recorded over multiple weeks, and stored in pcap files. From these files the timestamp, IP-addresses, and ports for the source and destination are extracted for each packet. Then, from this traffic we create ten sets each for HTTPS and SSH traffic, where each set represents a weekday from 9 AM to 5 PM. This restriction allows for reasonable consistency in using similar days and times, but still gives variety in the data, by having different (amounts of) users occupying the network. Outside of office hours and during the weekend there are less people on the network, and thus the traffic will become very sparse. This will result in inconsistent datasets and will make detection training difficult, since any training that is done on traffic during the day will not represent the night, and vice versa.

All sets are split further into a training set, a model set, and an injection set, which are 10%, 10%, and 80% by time of the original set respectively. The training set is used by the detection methods to train on legitimate traffic, and the model set is used by the covert channels to create models of legitimate traffic, as well as a source of IPDs for transmission. The injection set is the traffic with injected covert channels, over which the evaluation is performed.

To create different network conditions for these datasets, we simulate the addition of jitter on the transmitted packets. The jitter is modeled as normally distributed random values with mean 0 and standard deviations ranging from 1 to 5 ms, that are applied to every packet. This estimation creates plausible scenarios for network conditions, which are much like what is obtained from real traffic in [16]. Since only the IPDs of subsequent packets are used, the mean of the random values will cancel out, and is therefore not an important factor in the simulated jitter. Thus next to the sets without any added jitter there are five variations of jitter, representing low to high network disturbance, which are applied to all three of the sub-sets after a covert channel has been injected. It should be noted that it is unlikely for a network to not have any form of jitter. The sets without jitter are used to show the difference in performance of detection methods in an ideal situation, where there is no network jitter, compared to more realistic network conditions.

To give an overview of the behaviour and differences between SSH and HTTPS traffic, we obtain metrics from the full day sets. For the flows with a minimum of 1000 IPDs, the amount of unique flows

	1	2	3	4	5	6	7	8	9	10
Flows	71	64	83	99	61	335	136	75	50	68
IPDs/Flow	85546	40312	34434	27265	23972	4423	6101	29253	31817	34907
IPD Mean	0.102	0.135	0.239	0.303	0.327	2.638	0.881	0.217	0.241	0.173
IPD Std	2.983	3.245	4.065	4.714	4.805	20.691	9.553	4.023	4.824	4.212

Table 3.1: SSH dataset statistics

	1	2	3	4	5	6	7	8	9	10
Flows	546	563	632	503	360	350	701	538	550	649
IPDs/Flow	5321	5584	4495	5290	6618	5590	4614	5228	6825	5354
IPD Mean	1.016	0.895	1.096	0.836	0.663	0.884	1.033	0.808	0.594	0.699
IPD Std	8.782	8.303	8.546	7.297	7.060	7.900	7.704	7.270	6.147	6.717

Table 3.2: HTTPS dataset statistics

are counted, together with the average amount of IPDs contained in these flows. The minimum is chosen from the lowest sample size of the detection methods, given in Section 3.4. Flows containing less IPDs than this sample size will not have influence on the results of the detection methods, so they are not taken into consideration. Next to these two metrics we also calculate the mean and standard deviation (given in seconds) of the IPDs in the flows, to give an overall idea for the distributions of the traffic. The statistics for all ten sets can be seen in Table 3.1 and 3.2, for the SSH and HTTPS datasets respectively.

From these statistics we can observe some distinct differences, between the recorded SSH and HTTPS traffic. The SSH data generally has a lower amount of flows, but with a higher amount of packets per flow, whereas this is the opposite for HTTPS traffic. In the case of the mean and standard deviation of their IPDs, these values are considerably lower for SSH, than for HTTPS. This means that the IPDs are on average relatively small and have a low amount of spread for SSH, and conversely the IPDs are relatively large with more spread for HTTPS. Both of these traffic types are fairly consistent in the observed statistics, with the exception of day 6 and 7 for SSH. For both of these days we note a large increase in the amount of flows, with a lower amount of IPDs per flow, as well as a significant increase in the mean and standard deviation. These differences may be caused by an influx in the amount of users contributing to SSH traffic, with divergent traffic behaviour. Another explanation could be an increased utilization of multiple flows by the users, where each flow would also have higher IPDs.

3.2. Detection methodology

To be able to effectively compare the performance of the detection methods, we need to utilize the same detection methodology for every method. We therefore change the general application for the detection methods, from how they are applied in their respective papers, while keeping the original functionality intact. Our methodology is based on the premise that we have no knowledge of which covert channel technique an attacker might use. Consequently, we avoid biasing the detection method training on any specific covert channel technique, by only training on legitimate traffic. While partially training on covert channel traffic might improve detection rates for that specific covert channel, it might also cause a decrease in detection performance for other covert channel techniques, and is therefore not generally applicable in a realistic situation. Next to this, we still might not know what parameters an attacker will use for a certain covert channel technique, and thus training on the wrong parameters will lead to a worse performance. Taking these points into consideration, our detection methodology is as follows.

First we apply the detection methods over the training set and calculate their scores over flows in this set. Then depending on the direction of threshold (lower, upper, or both), a threshold value is determined for the detection method for that specific set. The threshold will determine if a sample of injected traffic is classified as being a covert channel, when it is over (or under) the given threshold for that detection method. The threshold value is taken from the calculated scores, with an allowance of 0.01 false positive classification. So, for example, if there are a 100 calculated scores for a detection method, the 99th or the 2nd value in the ordered scores will be set as the threshold, for upper or lower

threshold directions respectively. The use of a set false positive rate allows for a certain amount of outliers (or covert traffic) in the training set, by filtering out these scores.

Second we inject the covert channel traffic into the injection set, for each covert channel technique and variation. In the case of MB-CTC, RMB-CTC, and LBtNP variations that are based on a model of traffic, the model set is used to create a statistical model of legitimate traffic for a set, from which the IPDs of the transmitted packets are generated. Other CTCs, such as JitterBug, DPOI, and TR-CTC, use the model set as a source of legitimate IPDs, that they use to transmit their covert data. In the case of the SCC and On-Off covert channel techniques, which do not require legitimate traffic or models to operate, the model set will not be used. We transmit 10 kilobytes of the same random covert data for each covert channel and variation to create new mixed sets of injected traffic, using the models or legitimate traffic where applicable. The covert data is transmitted over a single flow using unique IP-addresses, which allows us to later distinguish the covert traffic from its legitimate counterpart. When using the same amount of data for each covert channel, the amount of time needed to transmit the data will differ per covert channel variation. For example, an On-Off covert channel with a time window of 10 ms will occupy roughly 13 minutes of traffic, whereas MB-CTC can span multiple hours, depending on the model it creates from legitimate traffic. So depending on which covert channel technique is used, it is unavoidable that this will cause part of the covert packets to go outside of the boundary of the time frame of the injection set. The downside from transmitting the covert data this way is that it creates a less realistic situation, where at points in the mixed traffic there are only covert packets. As a consequence from this we cannot realistically perform MB-CTC and RMB-CTC with refitting of the model with live traffic, since at a certain point in transmission there are no longer any live packets to do the refitting on. Despite these downsides we chose this method since the equal amount of transmitted covert data gives a fair comparison of detection on different CTCs, whereas there would be a lot less results to base the classification on for some CTCs when fitting the transmitted information inside of the time frame.

Lastly, we apply the detection method on each flow of the mixed set of legitimate and covert traffic. The threshold determined in the first step will then indicate for each sample in the flows whether or not it is classified as a covert channel. Then, using these classifications, and the knowledge of which samples were actually covert traffic, we can determine the false and true positive rates for the detection method on the covert channel variation. We define the false positive rate as being the percentage of legitimate traffic that is incorrectly classified as a covert channel traffic, with regards to the total amount of legitimate traffic in the set. Conversely, the true positive rate is defined as the percentage of covert channel traffic that is correctly classified as such, in relation to the total amount of covert traffic. Both the false negative and true negative rates can be easily obtained at each point from these two values, by subtracting the true positive rate and false positve rate, respectively, from 100 percent. This follows logically from the definition of these values, where the false negative rate is the percentage of incorrectly classified covert traffic, and the true negative rate that of correctly classified legitimate traffic. Since these other two values are somewhat redundant, we have thus chosen to only show the false and true positive rates to not clutter the resulting figures. Then, the false and true positive rates are obtained for each of the 10 sets of different days, from which the reported rates are then given by the average of these sets. The three-step process is performed for each combination of detection method and covert channel variation, to obtain the rates for every combination.

To show the effect of jitter on the detection methods and covert channels, we repeat the process for the different amounts of jitter as well. The jitter is first applied to the training and modeling sets, so that both the detection methods and covert channels use jittered traffic in the first two steps. This will lead to different thresholds for the detection methods, as well as different models, and legitimate traffic for covert channels. Then, after the covert data has been injected, the mixed set has jitter applied to every packet, to simulate the network conditions.

3.3. CTC settings

As discussed earlier, one of the difficulties in detecting covert channels is that it is unknown what type of covert channel technique an attacker chooses, as well as which values such a technique might use. It is therefore important to apply the detection methods to a broad range of settings for the covert channels. The settings that are used are plausible values an attacker might use, that are derived (and expanded upon) from the respective papers these techniques are discussed in. The different values generally

have a trade-off in transmission speed, robustness against errors from network jitter, and (in some cases) add a form of detection avoidance. For example, larger time windows will decrease the overall transmission speed of the covert channel, but network jitter has a smaller impact on the performance. The parameters for the variations of the covert channels are as follows:

- For the SCC the binary form will be used with τ between 10 and 100 ms, with 10 ms increments, where the two symbols are sent with delays $t_1 = \tau$ and $t_2 = 2 * \tau$.
- On-Off CTC will use the same 10 to 100 ms range as SCC for its time window.
- DPOI will use a 50 to 300 ms range, with 50 ms increments, for the covert time delay T_{ct}.
- The JitterBug covert channel uses the values 2, 5, 10, 15, 20, 25, 50, 75, and 100 ms for its timing window w.
- MB-CTC and RMB-CTC will be run without refitting and determine the parameters for their models from distribution fitting on the model set. Next to this another variant is added, where the models are fit on the IPDs that are smaller than one second. This will increase transmission speeds, but has the trade-off of creating models that are less accurate to the legitimate traffic.
- For TR-CTC the binary matching form will be used with the cutoff value τ_{cutoff} as the median of the IPDs, determined from the model set it uses to send packets. TR-CTC also includes a variant where the used IPDs are less than one second.
- The basic variant of LBtNP will send 8 bits with 3 packets, with Δ 10 or 50 ms, and δ 5 or 10 ms. For each combination of Δ and δ the additional random delay option will also be applied, doubling the number of variations. For the model variant 8 bits will be transmitted with 2 packets, and the model parameters are derived from the model set, which will include a low IPD version as well.

3.4. Detection method settings

Contrary to how the covert channels are varied in their parameters, the detection methods are chosen to be run with only one setting. In a real network setting the detection methods are likely to be setup to run with only one setting, while attackers can more easily change the operation of the covert channels they use. Also, without an effective way to select the settings for each detection method depending on the network, we cannot know beforehand which settings are ideal for that network. However, this does ensure a form of general applicability, by measuring the results without overly tuning the detection method settings on the specific traffic. Thus the detection methods will be applied with the recommended settings from their respective papers, or the settings that gave good results will be chosen, when there is an absence of recommended settings. All detection methods discussed in Section 2.2 will be evaluated, with the exception of the SVM and Arimoto-Blahut methods. This is due the fact that to effectively test the performance of SVM we would need to evaluate it on different amounts and combinations of the detection methods. Due to the amount of possible combinations, this falls outside the scope of this research. For Arimoto-Blahut, the way the data is obtained, it is not possible to calculate the symbol distribution using the original send time and the delay that is added from sending it over the network. Therefore this method is not suitable for detection under our current experimental evaluation. The settings for each detection method are as follows:

- Regularity uses a sample size of 2000, with a subsample size of 100, and a lower threshold direction.
- ε-similarity uses a sample size of 2000, and seven ε-values are used as a threshold. The first six are used as a comparison of smaller than or equal to the similarity value, with the values being 0.005, 0.008, 0.01, 0.02, 0.03, and 0.1. The last value is used as a comparison of greater than the similarity value, with its value being 0.1. The first six values use an upper threshold direction, and the last a lower threshold direction, with a majority vote deciding the classification of the sample being a covert channel.

- Mean-IPD uses a sample size of 2000, a bin size of 10 ms, and an upper threshold direction.
- Compressibility uses a sample size of 2000, and an upper threshold direction.
- The Kolmogorov-Smirnov test uses a sample size of 2000, a training sample size of 4000, and an upper threshold direction.
- First order entropy uses a sample size of 2000, and a lower threshold direction. The distribution for the binning is fitted from the training set, with a total amount of 65536 bins.
- Kullback-Leibler divergence uses a sample size of 1000, with a training sample of 2000, and an upper threshold direction. The bins that are used are equally sized in 1 ms, with the maximum amount being 300 seconds.
- Corrected conditional entropy uses a sample size of 2000, and a two-sided threshold direction. The distribution for binning is fitted from the training set, with a total amount of 5 bins.
- Welch's t-test uses a sample size of 2000, with a training sample size of 10 percent of the training set, and a lower threshold direction.
- For each of the three detection methods of kurtosis and skew, the used sample sizes are 2000, with a subsample size of 100, and a lower threshold direction.
- The non-parametric tests of Spearman-Rho, Wilcoxon Signed-Rank, and Mann-Whitney-Wilcoxon all use a sample size of 2000, with a step size of 200 IPDs. The metric is calculated from 10 p-values for Spearman-Rho, whereas Wilcoxon Signed-Rank, and Mann-Whitney-Wilcoxon use 15 p-values. All three detection methods have an upper threshold direction.

4

Results

In this section we evaluate the results obtained from performing the detection methodology introduced in Section 3. This is done on the basis of the true, and false positive rates of each detection method on each covert channel technique, and their variants. Due to the large amount of results, we choose some examples from these scores to exemplify where the strengths and weaknesses of each detection method lie. Then for each example we give an in-depth analysis of why these differences in performance occur. These analyses on the performance will help further the discussion on if these detection methods are applicable in a realistic scenario. The full results of performing the detection methods on each covert channel can be found in the appendix. Next to these analyses, a general overview is given for the performance of all the detection methods at the end of this section.

4.1. Entropy, Kullback-Leibler, and Corrected Conditional Entropy

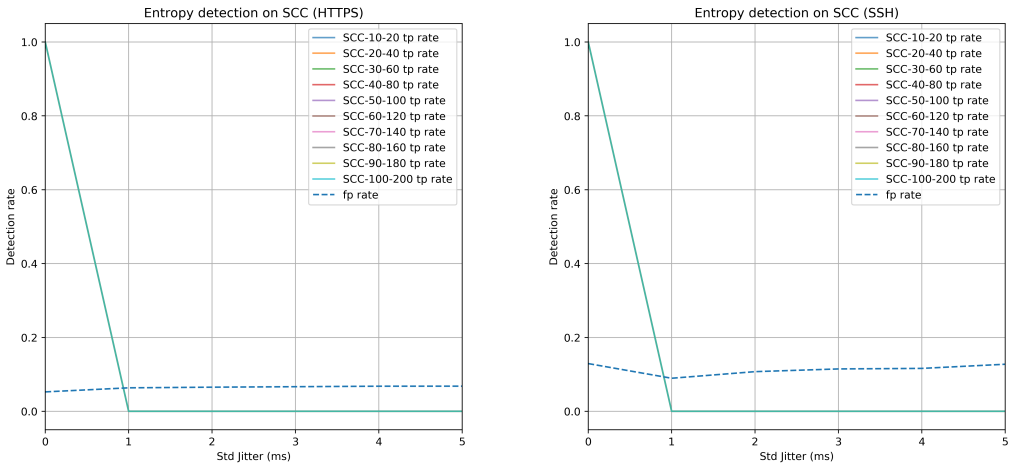


Figure 4.1: Entropy detection on SCC

The first detection method we will look at is the first-order entropy, for which the true and false positive rates can be seen in Figure 4.1. This detection method has true positive rates close to one at no added jitter and drops to zero for any of the used amounts of jitter. To explain why this occurs a more detailed examination is done on the entropy scores of the SCC variation SCC-10-20 for no jitter and 1 ms added jitter, which are given in Figures 4.2 and 4.3 respectively. When there is no added jitter the entropy scores of this particular covert channel are very close together and can be properly distinguished from the scores of the legitimate traffic. At 1 ms of added jitter, this distinction cannot be made, because the entropy scores come much closer to legitimate traffic.

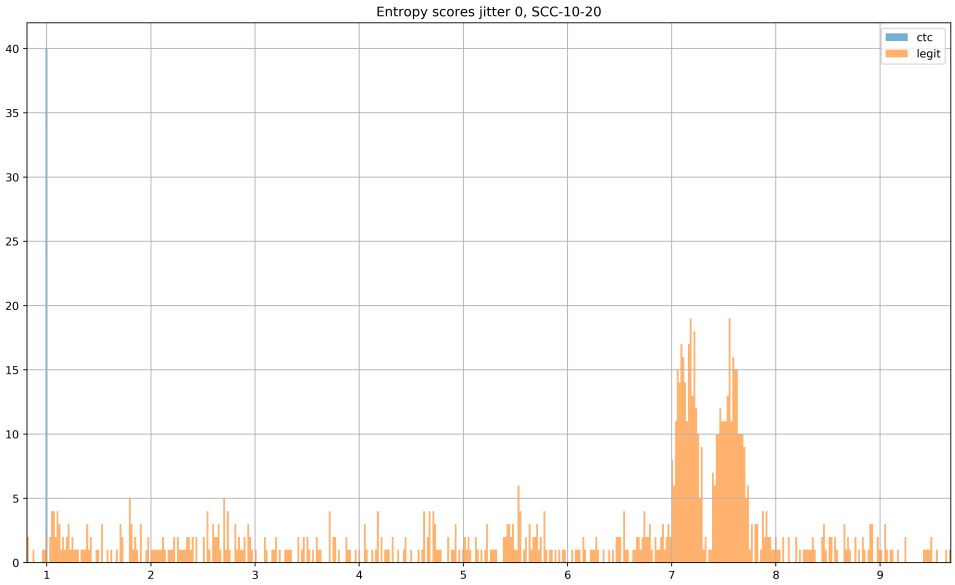


Figure 4.2: Entropy scores with 0 ms std jitter on SCC-10-20

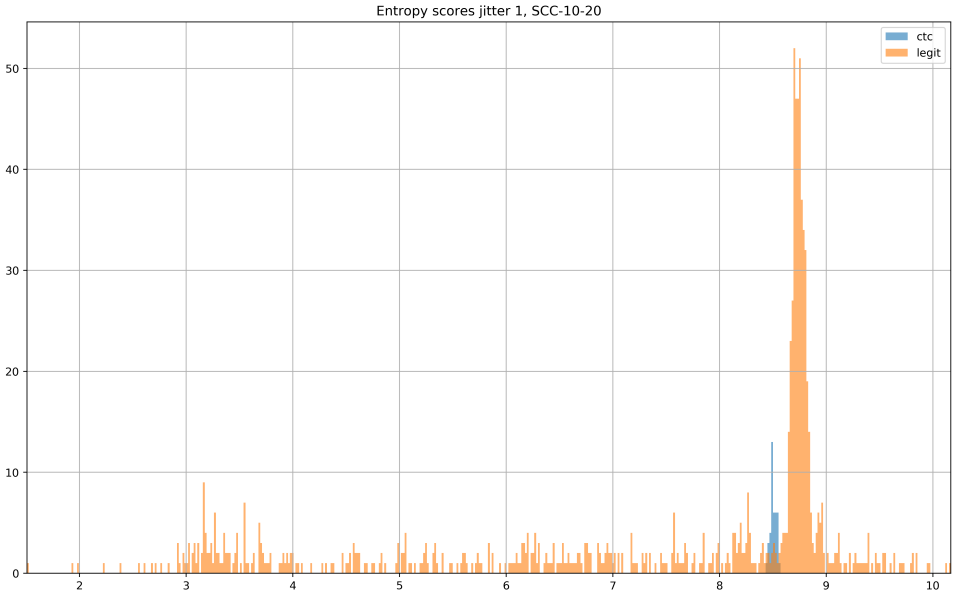


Figure 4.3: Entropy scores with 1 ms std jitter on SCC-10-20

As discussed earlier in Section 2.2.3 the entropy detection method is performed by sorting a sample of IPDs into bins according to the fit distribution from legitimate traffic, with higher resolution (i.e. more and smaller bins) around IPDs with higher occurrence. Then from these filled bins the entropy score is calculated over all of the bins according to the formula for entropy: $H(X_1,...,X_m) = -\Sigma_{X_1,...,X_m} P(X_1,...,X_m) log_2 P(X_1,...,X_m)$. When there is no jitter the transmitted IPDs for every varia-

tion of the SCC have exactly two possible values, which are then sorted into exactly two bins as well. A random string of transmitted bits will fill each of these bins close to evenly, which in turn gives similar scores for each sample of SCC traffic (resulting in the singular spike in Figure 4.2). The addition of jitter spreads out the possible values to the bins around the two original values, increasing the entropy score to the point where they are similar to legitimate traffic scores. A point to note is that this method does not take into consideration the positions where the bins are filled for a sample. For example, the same number of bins that are filled up to the same amount for two different samples will result in the same score, independent of if the first bins are filled or if they are anywhere else in the possible range of bins.

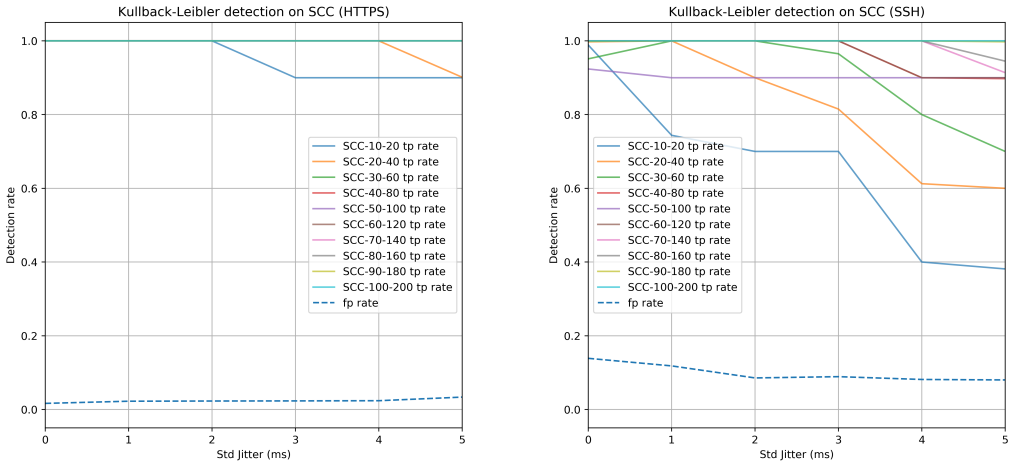


Figure 4.4: Kullback-Leibler detection on SCC

In comparison, the Kullback-Leibler divergence (KLD) metric discussed in Section 2.2.5, which is based on the relative entropy, does perform well under varying amounts of jitter. The true and false positive rates for KLD in Figure 4.4 show that there is a high detection rate for every variation of the SCC and jitter, with a minor performance drop for the SCC variants that use smaller IPDs. Much like the first-order entropy detection method KLD also works with binned IPDs, but instead uses bins of equal size and fills them beforehand with the value 0.5. Next to this, a binned sample from legitimate traffic is taken for comparison against samples during the detection process. The KLD score is then computed over all bins using $D_{KL}(P||G) = \sum_x p(x) \cdot log(\frac{p(x)}{g(x)})$, with $p(x)$ and $g(x)$ the probability at bin x of the traffic and training sample respectively. So if bins in a position are equally probable or have no occurrence in either sample, the resulting value of zero leads to no change in the score. On the other hand when there is a difference in probability for these bins, then the change in score follows the difference of the probabilities. In the case of the SCC with no added jitter the two possible values fill only two bins, so many bins have differing probabilities compared to the training sample, which leads to an increase in the KLD score. The addition of jitter does spread out values over more bins, however mainly for the cases such as SCC-10-20 and SCC-20-40 where there is more overlap between the IPDs of the CTC and training sample does this cause a significantly lowered KLD score closer to that of legitimate traffic, which then leads to misclassifications. Therefore the aspect of comparing positions of bins from the training sample and the traffic under analysis in KLD causes it to still perform well under increasing jitter, whereas the first-order entropy scores for the SCC increase to the point where these scores and the scores for legitimate traffic cannot be distinguished sufficiently.

The Corrected Conditional Entropy (CCE) method, discussed in Section 2.2.3, takes a different approach to determining the complexity of the examined traffic. Instead of the overall entropy, or relative entropy depending on position, the CCE looks at the differences in entropy between subsequent pattern lengths. This method is an approximation of the entropy rate which, unlike the actual entropy rate, can be applied to a finite series. At each level of pattern lengths within a sample, the CCE is calculated from the difference in entropy between two subsequent levels, corrected by the relative amount of unique

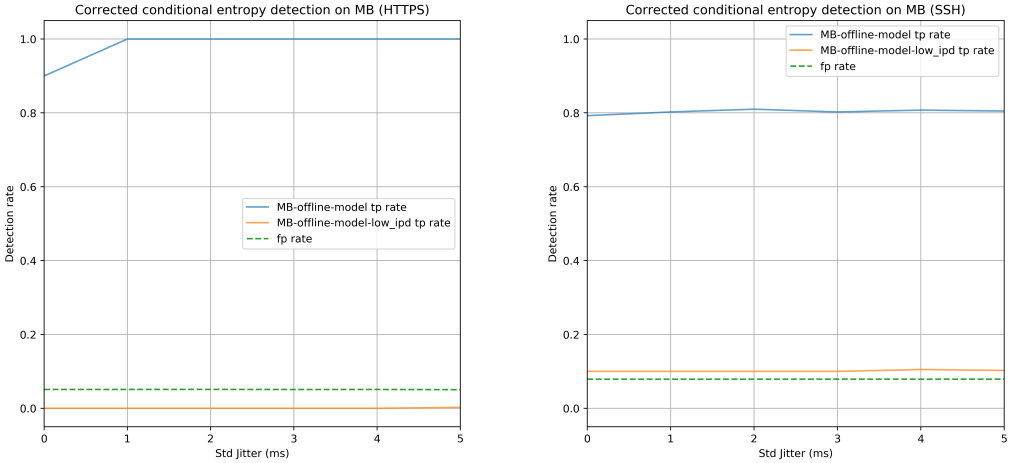


Figure 4.5: Corrected conditional entropy detection on MB-CTC

patterns of that length. This is given by the following formula:

$$CCE(X_m|X_{m-1}) = EN(X_m) - EN(X_{m-1}) + perc(X_m) \cdot EN(X_1)$$

The approximation of the entropy rate, and thus the resulting score, is then the minimum CCE value of all the pattern lengths in the sample. For traffic with highly regular patterns, using a small amount of values, the difference in entropy will be small between levels. Next to this, there are likely few unique patterns at each level, which will give a low overall CCE score. On the other hand, traffic containing more random patterns that utilize a broader range of values will have larger differences in entropy between subsequent pattern lengths. This type of traffic is also more likely to have more unique patterns, resulting in a higher CCE score. Legitimate traffic is assumed to lie somewhere between these examples, which leads to the usage of two thresholds.

From applying the CCE detection method on the CTC techniques we found that it most notably performs well on the covert channels that are based on models of traffic, which are MB-CTC, RMB-CTC, and the model variant of LBtNP. This performance can be seen in the detection scores for MB-CTC in Figure 4.5, where high true positives are achieved for both traffic types. However, from this figure we can also see that the performance is much lower on the low IPD variant, for which the model is based only on legitimate traffic IPDs of less than a second. This difference in performance occurs for all low IPD variants of the three CTC techniques mentioned above. One positive aspect that can be seen from the figure is the low impact of network jitter on the CCE detection method. The similarity in detection rates for different amounts of jitter is mainly caused by the employed coarse binning strategy. Due to the coarse binning the addition of jitter is less likely to change the bins the IPDs in the sample are placed into. This will then result in similar overall patterns for all variations of jitter, as well as similar scores. Further we note middling to high detection rates for DPOI and JB for the SSH traffic type, with no other significant results for the other CTC techniques.

The CCE scores for legitimate traffic in relation to the two variants of MB-CTC, SCC, and On-Off can be seen in Figures 4.6 and 4.7, for HTTPS and SSH respectively. From these figures we can discern that the full model variant of MB-CTC has relatively high CCE scores, when compared to legitimate traffic. These differences in scores are caused by how MB-CTC creates its IPDs from the model. The IPDs that are created are similar in occurrence to that of legitimate traffic, but not in the order in which they occur. This means that MB-CTC will have more varied and unique patterns spanning the entire range of the model, leading to a higher CCE score. This higher score is then sufficient to distinguish this variant of MB-CTC from legitimate traffic. In the case of the low IPD variant we can see that the scores are significantly lower than the other variant. These scores are much closer to those for both legitimate traffic types, making it more difficult to differentiate them, which leads to the low detection rates we have observed. The main cause of the lower CCE scores is the restriction on the possible values produced by the model, from basing this model on a smaller range of IPDs. The restriction on the values causes

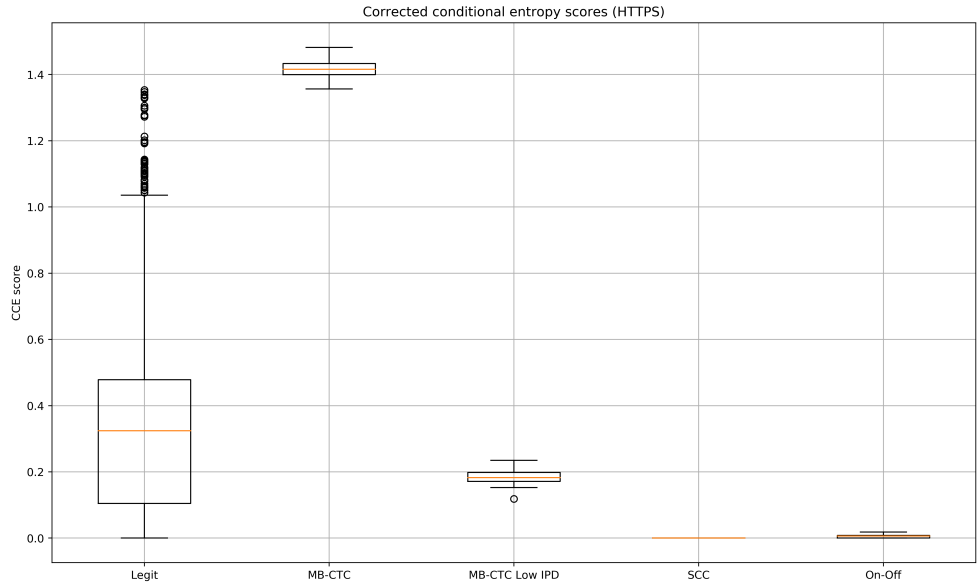


Figure 4.6: Corrected conditional entropy scores HTTPS

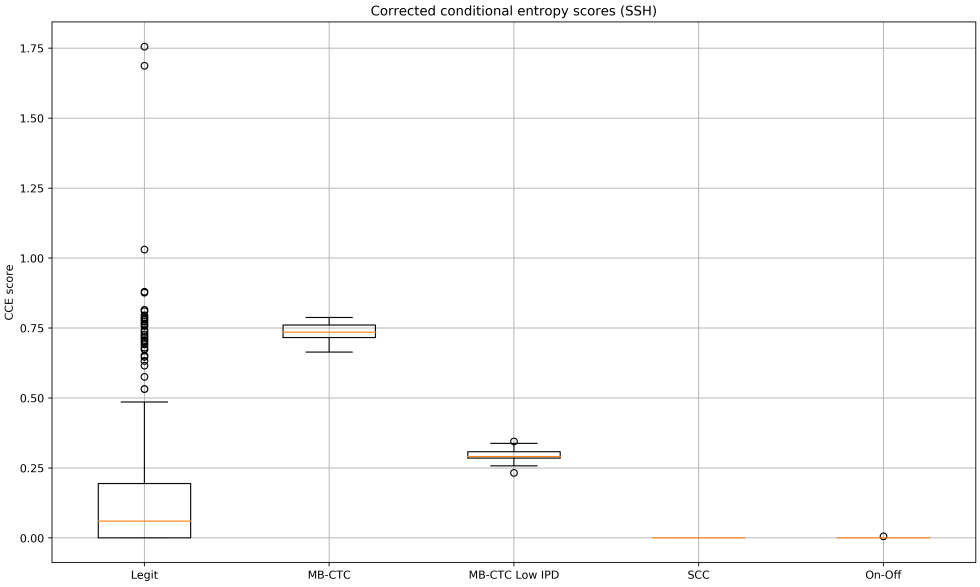


Figure 4.7: Corrected conditional entropy scores SSH

a smaller amount of bins to be filled, which are based on the full range of legitimate traffic, lowering the entropy at each level of pattern lengths. By placing the same amount of IPD patterns over such a smaller range of bins will also likely produce fewer unique patterns. Both of these factors then result in an overall lower CCE score compared to the full model equivalent of this covert channel technique, and thus the (partial) avoidance of detection. Another point to be noted from the figures is that the

CTC techniques that are less complex, such as SCC and On-Off, have CCE scores that are closely grouped together around zero. These scores are to be expected from the limited amount of values and regularity in the patterns that are used by these types of CTCs. However, for the legitimate traffic there are also a number of samples that produce low CCE scores, causing some overlap. Due to this overlap these covert channel techniques are not being detected well by CCE. The detection for less complex CTCs can thus theoretically be improved by utilizing a larger training threshold percentage at the lower side, with the trade-off being an increase in overall false positives.

4.2. Mean IPD

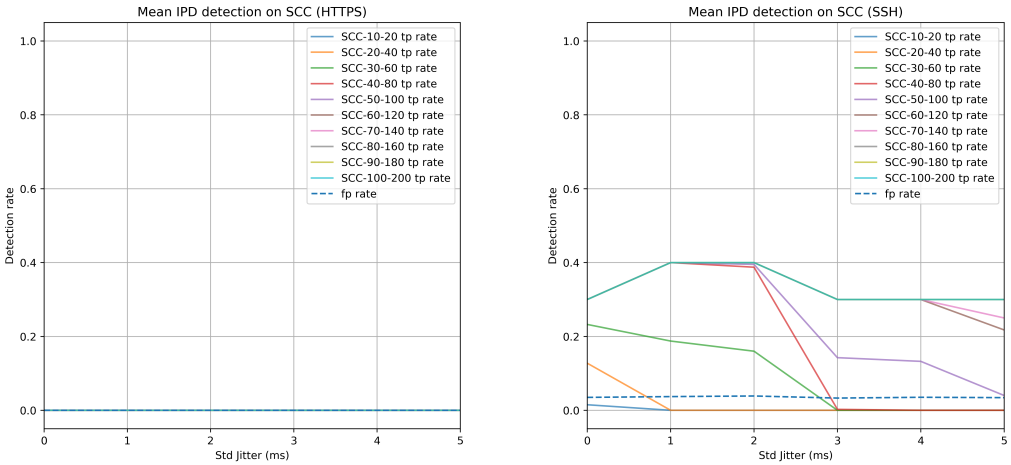


Figure 4.8: Mean IPD detection on SCC

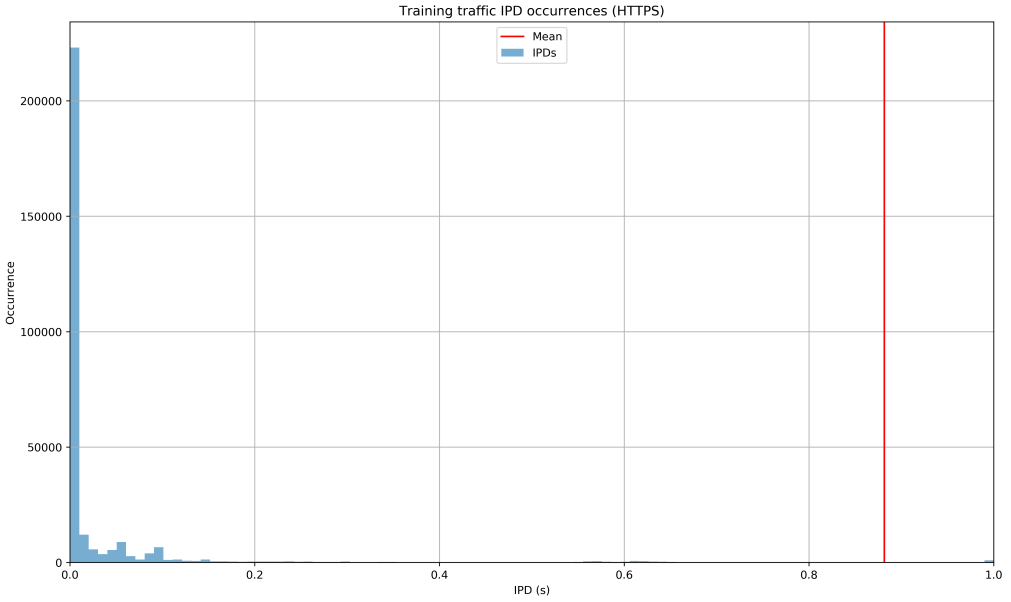


Figure 4.9: Training traffic IPDs HTTPS

Intuitively the mean IPD detection method should in theory perform well on the SCC technique of

sending information. From the two possible values there are also only two peaks of occurrences around those values, with the mean in between the peaks having little to no occurrences. The score obtained from $P_{CovChan} = 1 - \frac{C_\mu}{C_{max}}$, with C_μ the count at the mean and C_{max} the count at the largest peak, should then be (close to) one. However, as can be seen in Figure 4.8 the detection rates are not as expected from the intuition. In the case of SSH traffic the SCC variations that use smaller values have lower detection rates. Due to the binning size of 10 ms that the mean IPD method uses, a variation such as SCC-10-20 will have all values binned in two bins next to each other. The mean that should lie between those values will then lie at one of these two bins, resulting in a score closer to zero. The presence of jitter will also cause other variations, that have no occurrences at the mean under no jitter, to have increased counts at the mean. This can decrease the detection score under the threshold, for which it is then misclassified as legitimate traffic.

Aside from this, the main issue why this detection method does not perform well is from the assumption that for legitimate traffic the mean of the IPDs lies around the point where the most packets are. In Figure 4.9 binned training traffic IPDs for HTTPS can be seen, with their mean indicated in red and values larger than one second truncated for clarity. Most of the IPDs occur between 0 and 10 ms, but due to larger IPDs in the legitimate traffic the mean is shifted to the right of the bin where the most IPDs occur. A small number of outliers of large values can change the mean in such a way that it is not close to the highest peak. In many samples of this training traffic the count at the mean is very small or zero, which in turn gives high mean IPD scores during training. From these scores the threshold is determined, which is thus set to a high value. For HTTPS this threshold is set to one (i.e. the maximum score) in all cases, so neither legitimate traffic or CTC traffic is being classified as a CTC. This means that for all different covert channels the false positive and true positive rates are zero, which is what can be seen in Figure 4.8 for the SCC.

A solution one might think of is to use the median for finding the values needed to make this detection method work. This does find values during training that are closer to the peaks of the most occurring IPDs. However, since there are little to no occurrences between the peaks of the SCC, it gives no guarantee that the median lies between those peaks, and is therefore not a suitable solution. Next to this, the detection method also has limited usability on only covert channel techniques that use an even amount of values. In the case of CTCs with an uneven amount of values the mean is likely to lie at one of the peaks of those values, making it difficult to detect using this method.

4.3. ε-Similarity

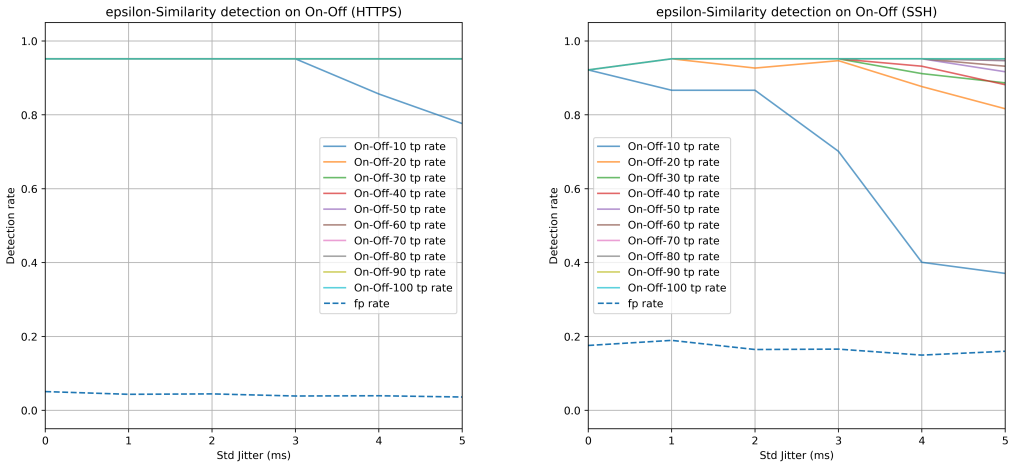


Figure 4.10: epsilon-Similarity detection on On-Off

Next we will look at the ε-similarity detection method we discussed in Section 2.2.1. This method takes a sample of sorted IPDs and calculates the relative differences, $|P_i - P_{i+1}|/P_i$, for each subsequent point in the sample. Then for six ε values the ε-similarity scores are given by the ratio of relative

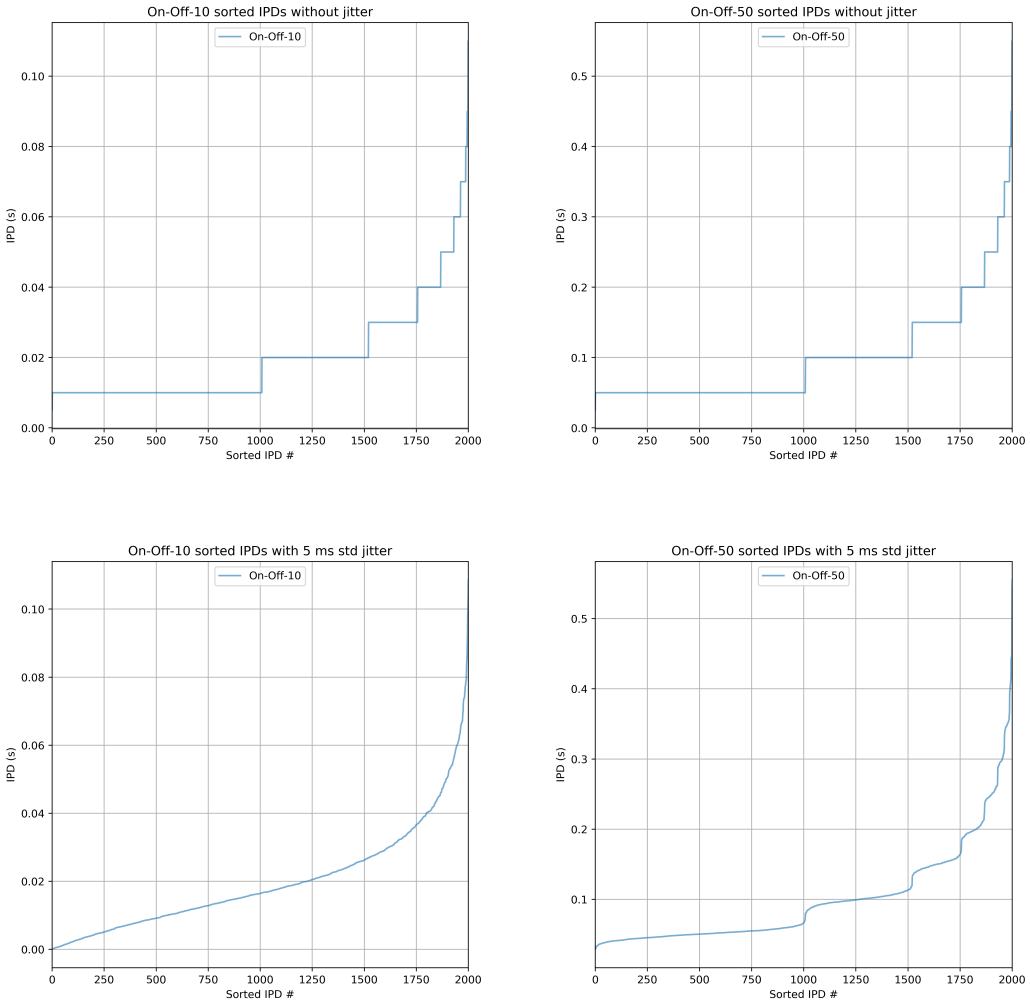


Figure 4.11: Sorted IPDs of On-Off traffic samples

differences that are lower than ε, with one more ε value that compares differences that are higher than that value. A majority vote of the scores compared to the thresholds then determines whether or not a sample is classified as a CTC. This detection method has high detection rates for all of the CTCs that use set values or time intervals, which are the SCC, On-Off, and the non-model variants of LBtNP. There is however a performance drop for the lower-valued variations of these CTCs, when there is an increase in jitter, which can be seen for On-Off-10 in Figure 4.10.

To examine why this performance drop happens, we will take a closer look at the intermediate process of sorting the IPDs. In Figure 4.11 the sorted IPDs of the variants On-Off-10 and On-Off-50 are shown, under no jitter and 5 ms standard deviation of jitter. For On-Off CTC a zero in the transmitted string leads to a set silence time interval where no packet is sent, and a packet is only sent in the middle of this period for a one. Then depending on the amount of zeroes before a one different combinations are formed, which will lead to distinct IPD values for those combinations. When there is no added jitter there are clear separations for both On-Off variants between the different combinations, where all IPDs of the same combination have the same value, creating horizontal lines and peaks at the transitions of combinations. With the addition of jitter the horizontal lines have become inclined and the points where transition occur are no longer visible for On-Off-10. In the case of the variant that uses larger time intervals On-Off-50, the lines are less steep and there is still a visible transition between the different combinations. The slopes with a larger incline will have higher relative differences between each subsequent point, and thus more relative differences that are larger than the various ε values. The

ratios for the similarity scores are then much more likely to be under the threshold, resulting in lower detection scores. Conversely, the less inclined slopes of On-Off-50 retain the low relative differences more than On-Off-10. These then lead to higher ratios of relative differences under the ε values, and thus still have a high detection score.

Next to this, the ε-similarity detection method has low performance on the other types of CTCs or variants that use legitimate traffic or models based on that traffic. These types of CTCs have sorted IPD samples that are too similar to legitimate traffic, resulting in similarity scores that are close to legitimate traffic as well. JB and DPOI add an extra delay to the traffic that they use, but due to the high amount of possible values in the legitimate IPDs, as well as the random values used in JB, this does not show in the similarity scores. In the case of TR-CTC there is no added delay, and the pre-recorded, legitimate traffic is merely played back in a different order. The sorting of the sample of IPDs by the detection method removes the order that TR-CTC introduces, and so the only information left is of the legitimate traffic IPDs. This therefore makes ε-similarity unsuited for detection of this particular covert channel.

4.4. Regularity

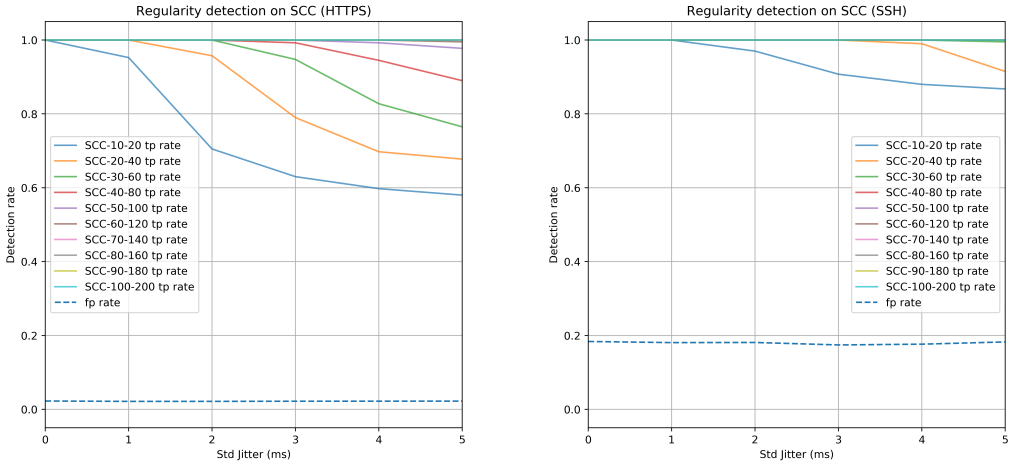


Figure 4.12: Regularity detection on SCC

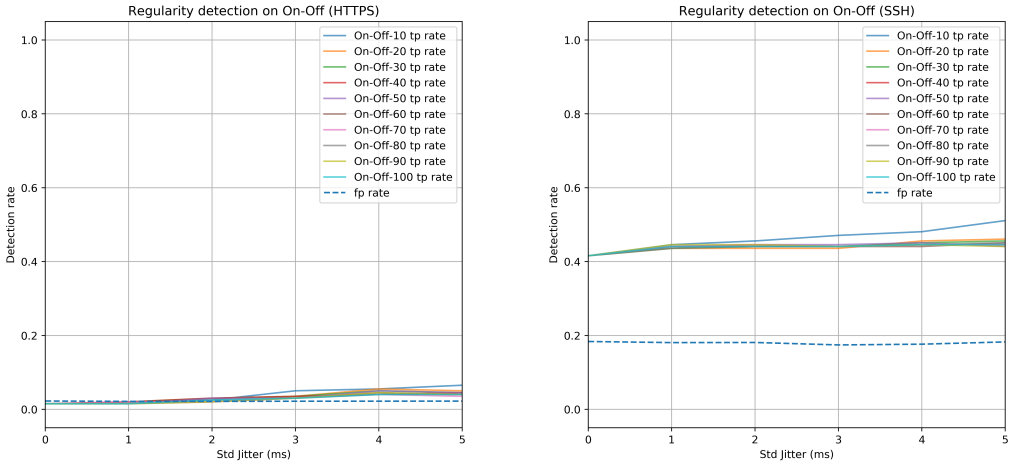


Figure 4.13: Regularity detection on On-Off

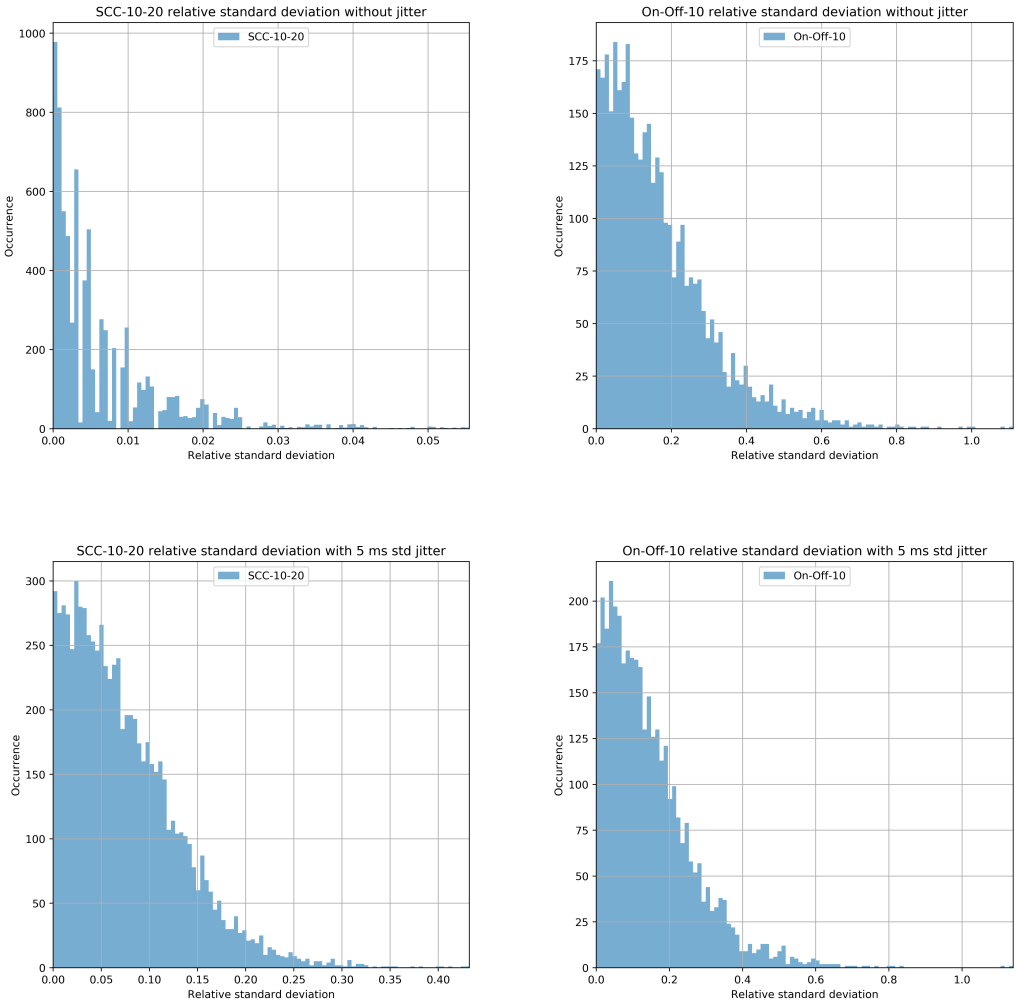


Figure 4.14: Relative standard deviation of SCC and On-Off traffic

In Section 2.2.1 we discussed the regularity detection method, for which the score is given by the formula $STDDEV(\frac{|\sigma_i - \sigma_j|}{\sigma_i}, i < j, \forall i, j)$. The detection method thus works by splitting a sample of traffic into sub-samples, and for each the standard deviation is calculated. Each of these values are then compared to all the standard deviations that come after it, which we will call the relative standard deviations. The regularity metric is then the standard deviation of all these relative standard deviations. This detection method has high true positive rates on the SCC and the non-model variants of LBtNP, with a middling performance for On-Off in SSH traffic. The detection rates can be seen in Figure 4.12 and 4.13, for the SCC and On-Off respectively. For the SCC there is a lowered performance for the variants that use smaller values, under increased jitter. Especially for HTTPS the decrease can be clearly seen, in order of the values that the variants use, where variants that produce smaller IPDs will have a larger decrease.

To explain why these issues in performance on the SCC and On-Off occur, we will look at the differences in the relative standard deviations for certain cases. In Figure 4.14 the relative standard deviations can be seen for samples of the variants SCC-10-20 and On-Off-10, under no jitter and 5 ms of added jitter. In the case of the SCC without jitter each sub-sample will only have different amounts of exactly two values, which lead to similar standard deviation values for all of them. Since the standard deviation of each sub-sample will lie very close together, the relative standard deviations will be small. So with a majority of small relative standard deviation values, the regularity score is then also low. In comparison with the case of 5 ms added jitter, the 10 and 20 ms IPDs are changed by such an

amount that there are larger differences in the standard deviations of each sub-sample. The potential differences between these values then lead to increased relative standard deviations and, following this increase, higher regularity scores. The change caused by the jitter can be seen in the figure, where no jitter has a maximum relative standard deviation of approximately 0.05, and 5 ms of jitter that of around 0.4. The standard deviations of sub-samples are also affected by jitter for larger-valued variants, but since the amount of jitter is relatively small compared to the values sent, this has a lesser impact on the change in relative standard deviations and thus the regularity score as well. As indicated earlier, the IPDs for sent packets in the On-Off covert channel depend on the amount of zeroes before a one in the transmitted string. Thus, depending on the string combination, this means that even for a variant such as On-Off-10 that uses a small time window, the IPDs can vary widely between each sub-sample. These outliers in the IPDs significantly increase a portion of the standard deviation values of sub-samples. Comparisons between small and large standard deviation values then lead to large relative standard deviation, which in turn produce larger regularity scores. There is also not a big difference in the relative standard deviation for added jitter, compared to no jitter. Since On-Off already has variation in its values, the jitter does not affect the scores by much. The figure shows that On-Off-10 has relative standard deviations, regardless of jitter, that are comparable to that of SCC-10-20 under 5 ms jitter, which translates into lower detection scores. So the main reason why regularity performs well on the SCC and non-model variants of LBtNP is that they use consistent values, without any (significant) outliers.

In the case of other covert channel techniques, that use real traffic or models, this detection method does not perform well. The legitimate traffic or model that is used for transmission of course has similar characteristics in variation and standard deviations as the traffic that the detection method is trained upon. When (small) values are added to the IPDs this does not have a significant impact on reducing the (relative) standard deviation of sub-samples, which will thus still be similar to legitimate traffic. Therefore this detection method does not have much success in detecting these covert channel techniques.

4.5. Compressibility

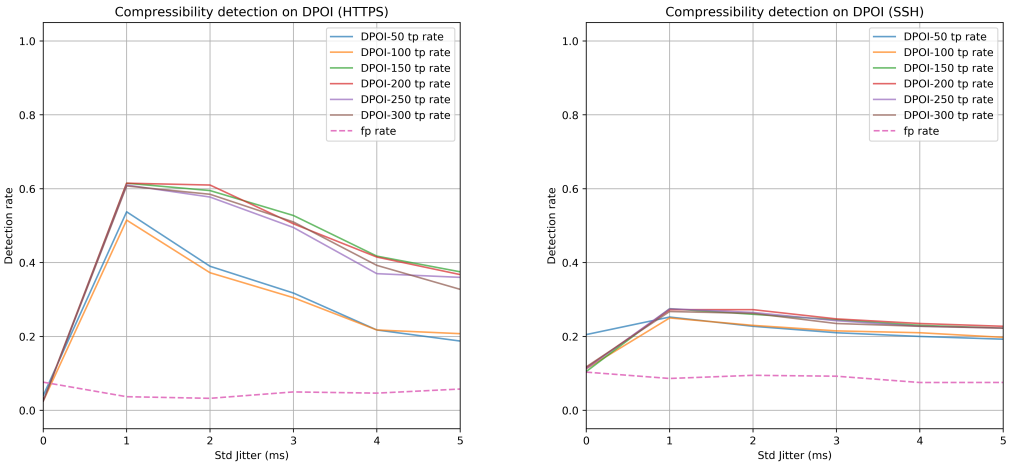


Figure 4.15: Compressibility detection on DPOI

The compressibility detection method discussed in Section 2.2.1 looks at the differences in compression lengths of IPDs for legitimate and CTC traffic. The method does this by taking a sample of traffic, and ignoring IPDs of length greater than or equal to one second, translates each IPD into a string. The string is created by taking the rounded first two significant numbers of the IPD and prepending these numbers with a letter, signifying the amount of zeroes after the decimal point. For no zeroes after the decimal point no letter is prepended, for a single zero the letter is an A, for two zeroes the prepended letter is a B, and so on. Then the string, from all the combined IPDs in the sample, is com-

pressed using an off-the-shelf compression algorithm, from which the compressibility score is given by the length of this original string, divided by the compressed string length. A compression algorithm will generally look for patterns and reoccurring values in the string to replace. This is why, for example, a string with certain pattern or many of the same values will be compressed into a much smaller size, compared to a string of random symbols. The same thing then also happens for the translated IPDs, where values close together will have the same prepended letter and similar numerical values, with a smaller compressed string, compared to that of values further apart. Therefore, CTC traffic with reoccurring values and/or patterns will result in a smaller compressed string, and should thus have a higher compressibility score, in comparison to that of the more varied legitimate traffic. In the case of HTTPS traffic the detection method has near full scores for the SCC, for all examined amounts of jitter, as well as high true positive rates for On-Off. Next to this, the method produces varying results for non-model variants of LBtNP, DPOI, and JB, depending on the used variants and the amount of jitter. From the detection scores two main issues arise with the compressibility detection method, where for these last three covert channel techniques the score at no jitter can be very low, and that SSH has scores that are distinctly lower than HTTPS. In Figure 4.15 these issues can be clearly seen for the detection on DPOI.

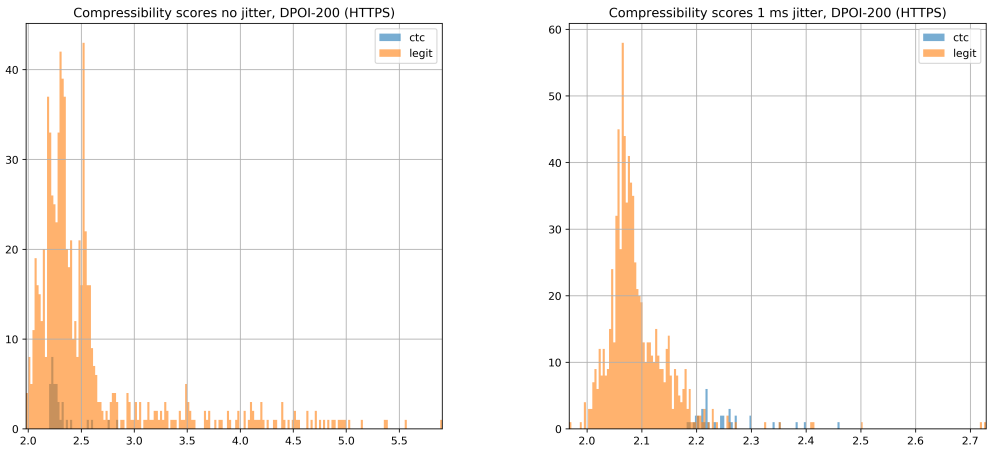


Figure 4.16: Compressibility scores on DPOI-200 using HTTPS

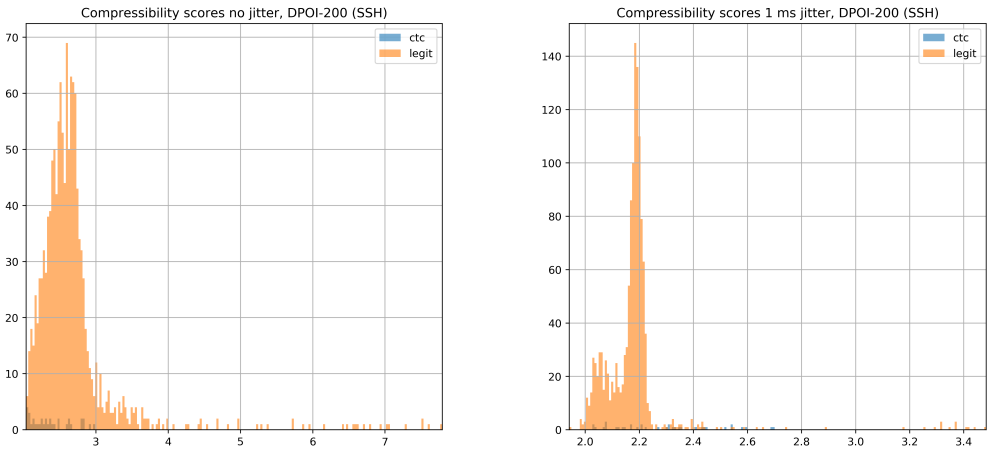


Figure 4.17: Compressibility rates on DPOI-200 using SSH

The low detection rates at no jitter are mainly caused by the higher compressibility scores from the training traffic. These higher compressibility scores come from more frequent occurrences of the same or similar IPD values. In Figure 4.16 we can see that CTC traffic scores have a majority that overlap with the most occurring legitimate scores, which thus results in many classification errors. When subjected to jitter, the legitimate compressibility scores are reduced, since the random values of jitter disrupt patterns and the similarity of IPDs. For DPOI the reduction in compressibility happens to a lesser extent. In Section 3, we have shown that HTTPS traffic on average consists of large values, with a significant amount of standard deviation. The process of compressibility then only takes the IPDs of this traffic that are under one second, reducing the amount of traffic under consideration. DPOI uses a set of IPDs to transmit its data and the HTTPS traffic that is used is made up of relatively large IPDs, compared to the added value of DPOI. This means that there is a certain amount of traffic that is not being considered by this detection method, since these IPD values are larger than one second. Next to this, the addition of a small amount of jitter is also less likely to alter the values of the strings, consisting of the prepended letter and significant numbers, obtained from the translated IPDs. Therefore, the compressibility scores are reduced less than that of legitimate traffic. With this difference in score change there is a more clear distinction between the legitimate and CTC scores, resulting in an increase in detection rates when a small amount of jitter is added. For the compressibility scores on SSH, given in Figure 4.17, the same score similarity can be seen under no jitter, but the distinction between the covert channel and legitimate traffic with 1 ms jitter is not as apparent as for HTTPS. Under no jitter SSH has higher compressibility scores on average than HTTPS. This can be caused by a higher amount of similar values, or by processes that send packets with a certain regularity. SSH in general consists of smaller IPD values than HTTPS, which in turn affect the IPD values produced by DPOI and their translated strings. With more similar IPDs between the DPOI and legitimate traffic, their scores are also affected similarly by the added jitter. Therefore, even with jitter their compressibility scores have a major overlap, resulting in lower detection rates.

Summing up, the compressibility detection method is successful for certain covert channels, but is dependent on the type of overt traffic that is used. For the HTTPS traffic that we used it can correctly classify the SCC and On-Off and, depending on the conditions of jitter in the network, certain others as well. However, in the case of SSH the detection scores are generally lower, and more sensitive to jitter.

4.6. Welch's t-test

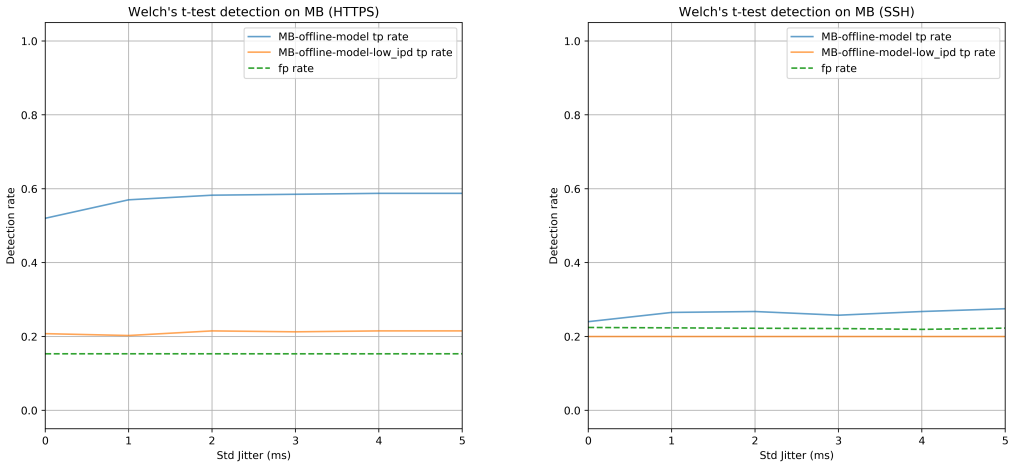


Figure 4.18: Welch's t-test detection on MB-CTC

In Section 2.2.5 the Welch's t-test is discussed, which is an adaptation of the student's t-test that allows for unequal size and variance of the two samples that are tested. The t-test is used to test for the null-hypothesis of whether the means of two populations are equal. Instead of rejecting or accepting the

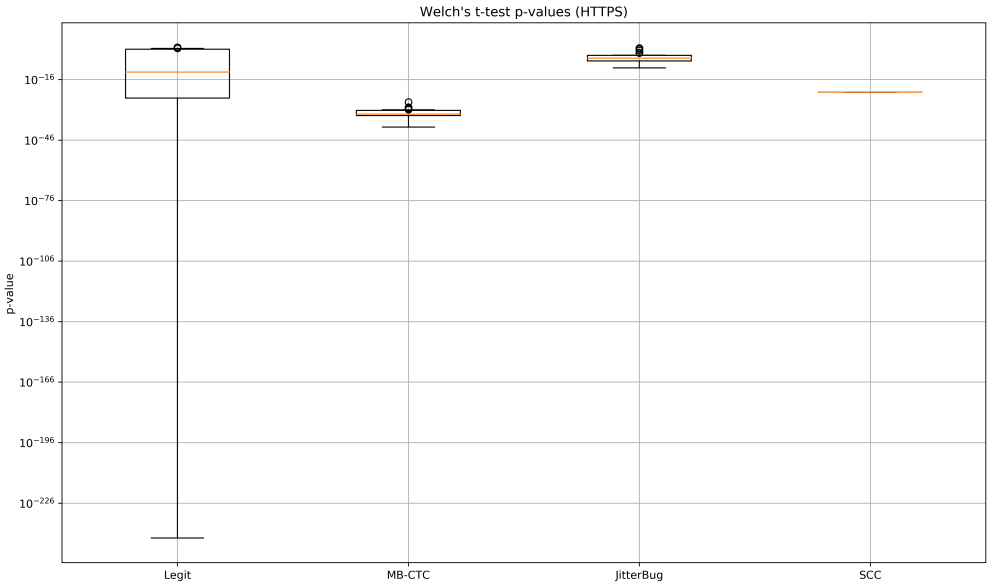


Figure 4.19: Welch's t-test p-values for legitimate traffic, MB-CTC, JB, and SCC

null-hypothesis, the p-values obtained from the test are used by the detection method as a threshold. A larger sample of legitimate traffic from training is compared to samples of observed traffic, to determine if these samples are generated by a covert channel.

The Welch's t-test detection method has decent scores on HTTPS traffic for the variants of covert channels that use a model closely resembling the legitimate traffic, meaning MB-CTC, RMB-CTC, and the model-variant of LBtNP, that use the full range of IPDs for their models. The detection scores for MB-CTC can be seen in Figure 4.18, for which the scores of the other two covert channel techniques are similar. From these scores it can also be observed that the detection method is robust against the effects of jitter. On the other hand, for the other covert channels and variations there are no notable results, with the maximum true positive rate being around 30 percent. Next to this, the higher false positive rate compared to the other detection methods makes it less usable in a realistic application, since the amount of legitimate traffic that is wrongly classified diminishes the effectiveness of the detection method. In Figure 4.19 we show the p-values (in the form of box plots) of all the observed legitimate traffic, compared to MB-CTC, JB, and the SCC, for one of the HTTPS sets. From these values we can observe that the covert channels have p-values that are grouped closely together, with no large outliers. The legitimate traffic, however, does have a wide range of possible values and outliers, that are derived from the different flows and behaviours present in the traffic. Of these three covert channels only MB-CTC has a majority of its values under that of the legitimate traffic, which leads to more correct classifications and thus increases performance. For the model-based covert channels the Welch's t-test captures the differences between values generated from a model and legitimate traffic, especially in the case of models close to the distributions of legitimate traffic. In the case of the other covert channels, their scores fall inside the range of legitimate traffic and are thus more likely to be wrongly classified.

In the original work where the Welch's t-test is evaluated [2], the scores obtained are substantially higher than the scores we present here. For the covert channels techniques of MB-CTC and JB the true positive rates are equal or over 82 percent, with false positive rates lower than or equal to 11 percent. The disparity in the detection results between our tests and those described in the original work could be caused by multiple factors. One of these factors might be the way the detection method is applied, in their methodology. The methodology is performed by injecting half of each training set with covert traffic, with the other half untouched, to represent legitimate traffic. Then, by applying the detection method

over both halves, the threshold value is optimized to yield the optimal classification rate, in terms of true positive and negative rates. Using the threshold value obtained from the training, the detection method is then performed against a test set, injected with the same covert channel techniques and parameters. The issue with this methodology is that it works under the assumption that it is known which covert channel techniques and parameters will be used by the attacker, and to train the detection method on this. This introduces a certain amount of bias into the detection, where the general performance on other CTCs and variations might not be similar. Next to this, training under the assumption of which covert channels and parameters are used, creates a less realistic scenario. Another possible factor for the difference in performance could be from the dataset that was used in the original work, which is network traffic from the Waikato I captures from 2005. Aside from the age of the dataset, which might not be representative of current network traffic behaviour, the data is likely dissimilar to the traffic we perform our tests on. The threshold value obtained from training in their work is given as 0.76, which is not close to the values we have obtained for legitimate traffic, as can be seen in Figure 4.19. This could indicate that the traffic is more homogeneous or similarly distributed than the data that we are working with, leading to more grouped and higher p-values for legitimate traffic.

Despite the lower performance demonstrated here, there is a potential for the Welch's t-test to perform well on many of the covert channel techniques. The p-values for covert channels are generally grouped close together with a small amount of outliers, which could make for good classification. However, the main problem that prevents it from performing well is the spread and amount of outliers of legitimate traffic p-values, from which the low true positive and high false positive scores stem. Perhaps with some adjustments to the detection method, that increases the overall scores for legitimate traffic and decreases their spread, this problem could possibly be solved.

4.7. Skew and Kurtosis

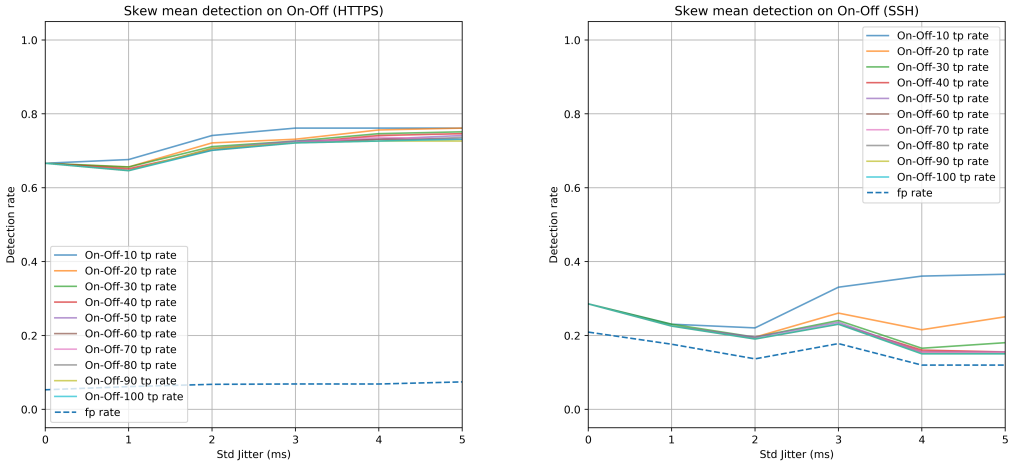


Figure 4.20: Skew mean detection on On-Off

The skew and kurtosis detection methods, discussed in Section 2.2.5, are higher statistical moments describing the shape of a distribution. For a distribution, the skew measures the asymmetry in the amount that the data is skewed within the distribution, while the kurtosis measures the amount of (extreme) deviation around the tail-ends of the distribution. For each of these shape metrics three detection methods are formed from calculations on the sub-samples, using the mean, standard deviation, and regularity. The detection methods based on the mean are the most straightforward application of the shape metrics, wherein the scores are directly related to the metrics. On the other hand, those based on the standard deviation and regularity observe variations between the skew and kurtosis values between sub-samples. The overall variation within the sample is measured by the standard deviation, and the variation between sub-samples over time is measured by the regularity detection methods.

For the detection methods based on skew and kurtosis we report significant results on the non-

model variants of LBtNP, On-Off, and the SCC. Under HTTPS traffic the skew mean method achieves 100 percent true positive rates for these LBtNP variants and the SCC, as well as the highest true positive rates for On-Off of around 70 percent. In comparison, for SSH traffic the detection methods generally perform worse. Notably with On-Off, the skew mean only achieves true positive rates between 15 and 35 percent for this type of overt traffic. In Figure 4.20 the difference in performance for the skew mean between the two types of overt traffic can be clearly seen for the On-Off covert channel. Next to this, the detection methods based on the standard deviation and regularity show a decrease in performance under higher amounts of jitter.

The overall performance on only the three named covert channel techniques is due to the set values that are used in them. From the set values, the distributions for these CTCs have distinct shapes that are dissimilar to that of legitimate traffic. Conversely, the shapes of the other covert channels (and LBtNP variations) are defined by the legitimate traffic or models they employ, which are thus similar to legitimate traffic and more difficult to detect using these methods. There is also a difference between the shapes of the distributions for HTTPS and SSH, defined by processes like user interaction, that create smaller values for SSH. As such, this variation of shapes for SSH prove to be more difficult to distinguish from covert traffic, compared to HTTPS. In the case of On-Off there is a greater possibility of outliers within each sub-sample, from the amount of zeroes before a one in the covert data. These outliers shift the shape of the distributions and increase the skew and kurtosis values, closer to those of legitimate traffic. Especially for the detection methods using standard deviation and regularity, the larger possible differences between sub-samples make it more difficult to detect On-Off traffic. Additionally, similar to what we have described in the discussion for the regularity detection method, the effect of jitter also applies for the skew and kurtosis regularity (and to a lesser extent for the standard deviation). Since the variation in the skew and kurtosis is used as scores for these detection methods, small changes in the IPDs due to jitter can affect these scores by a significant amount, depending on the IPD sizes used by the CTC variants.

In comparison to the original work [2], we do not attain similar scores for the covert channels that they discuss. Note that this work is the same as the one for the Welch's t-test, so we argue that the concerns regarding the dataset and methodology discussed in that section apply here as well. An example for one of these concerns is that although their methodology achieves decent true positive scores, there are instances where the false positive rate goes up to 50 percent. The relation of this false positive rate is such that, for similar network conditions, half of the overt traffic would be incorrectly classified as covert. With networks that have a large amount of overt traffic, relative to the possible amount of covert traffic, this would provide a significant number of absolute false positives to work through. Thus, in realistic scenarios where half of the legitimate traffic is classified as covert channel communications, the relevance of correctly classifying covert traffic is diminished by a substantial amount. Aside from these concerns, we find that the detection methods based on skew and kurtosis can perform well on select covert channel techniques. However, we cannot evaluate the capabilities fully for these methods, since the authors intended to apply them as classifiers for the use in SVM, which is outside the scope of this thesis. There could be combinations of the scores from these detection methods, that are distinct for both overt and covert traffic. These combinations would possibly allow a machine learning method, such as SVM, to correctly classify other CTC techniques, for which the individual detection methods show low performance. Thus, even though the detection rates of these singular detection methods on these CTCs is low, the use of SVM could improve performance when applied with a combination of these methods.

4.8. Kolmogorov-Smirnov

In Section 2.2.3 we discuss the Kolmogorov-Smirnov (KS) test, which measures the largest distance between a reference distribution and an empirical distribution function, or between two empirical distributions. For the use in covert channel detection, a training sample of legitimate traffic is used as a fingerprint and is compared against samples of possible covert traffic. Large KS-test scores between the empirical distributions of these two samples can indicate the existence of covert channel traffic, while smaller scores are more likely to be obtained from legitimate traffic.

For HTTPS we report near full true positive rates under no jitter for the SCC, On-Off, and the non-model variants of LBtNP. With the addition of jitter there is a significant decrease in these rates, depending on the size of IPDs used by these covert channels. The variants that use smaller IPDs show

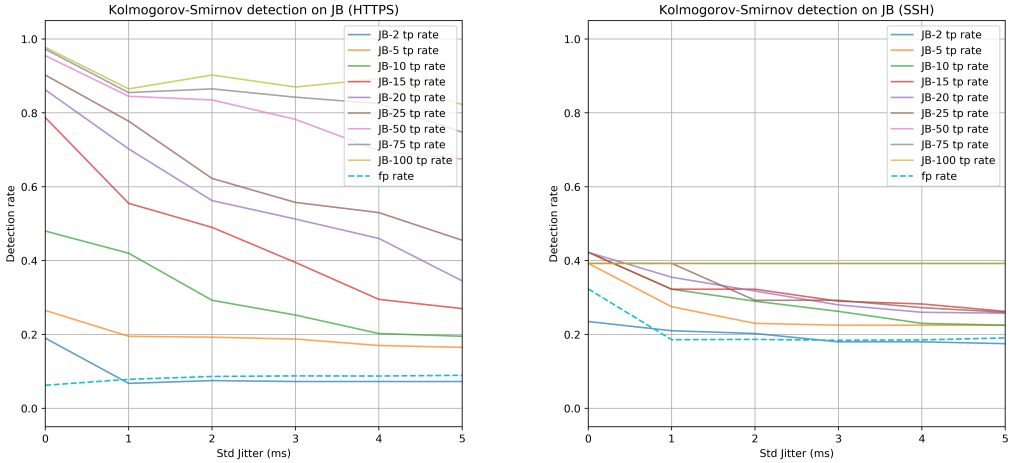


Figure 4.21: Kolmogorov-Smirnov detection on JB

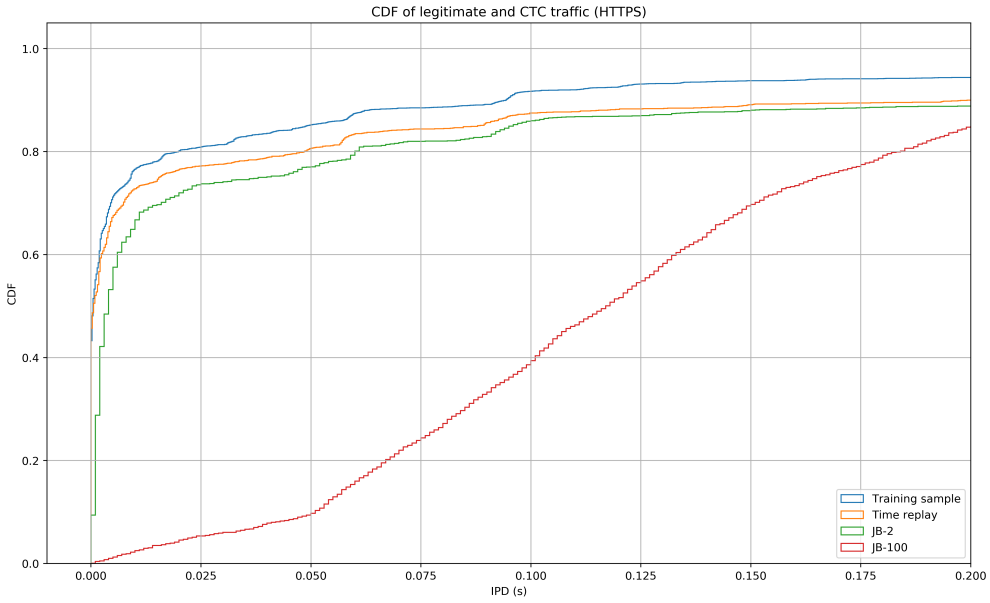


Figure 4.22: CDF of legitimate, Time Replay, and JitterBug traffic IPDs

a steeper decline in their detection rates, with an increasing amount of jitter. For the CTCs that use legitimate traffic or models we find no significant results, with the exception of JB. In the case of JB with HTTPS traffic there is a wide range of detection rates, depending on the variant that is being detected. With the variant that uses the largest w-value of 100 the true positive rates lie between 80 and 100 percent, while these rates are around 10 or 20 percent for the smallest JB variant. Next to this, the performance on SSH is generally worse, with false positive rates of more than double that of HTTPS. In Figure 4.21 the varying detection scores can be seen for the KS-test on JB, as well as the performance difference between HTTPS and SSH.

The variation in detection performance is mainly caused by the dissimilarity in shapes of the empirical distributions, either between the training sample and covert channels, or between different samples of legitimate traffic. As we have shown in Section 3, the distributions for SSH are fairly consistent,

with overall low values and a relatively small amount of standard deviation. There are, however, two datasets that greatly deviate from these values. This indicates that there could then also be larger variance in distribution shapes for the different flows of SSH, within the training set, as well as the injection set. These differences in shapes affect the capability of detecting covert channels, and due to dissimilarity in the training sample and injection set flows, there is a large increase in false positive rates. Under no jitter the CTCs with a limited amount of IPD values, such as SCC, On-Off, and LBtNP, will only have peaks in their distributions at those values, with flat lines connecting them. The maximum distance between the distributions for these covert channels and the training sample will thus likely be great, and can be easily distinguished from legitimate traffic. With the addition of jitter, however, the transitions from each peak of distinct values will be more gradual, which decreases the maximum distance to the training sample. This smoothing in transitions occurs more for variants that use smaller values, with increasing jitter, as well as that the size of IPDs that they use are closer to legitimate traffic. Both the smoothing and size of used IPDs contribute to the lowered detection performance on these types of covert channel techniques. In the case of JB the IPDs are based on legitimate traffic, which generally follows the distribution of the training sample. JB adds delays to the legitimate traffic, to achieve certain values, which shift the distribution of the original traffic, determined by the w-value used. To mask the distinct values that this creates, a random jitter is applied to each IPD, which is based on the w-value as well. Thus, depending on the amount that the distribution is shifted, the dissimilarity in the distributions increases and this reflects in the maximum distance and detection scores accordingly. In comparison to the other CTCs that use models or legitimate traffic, the KS-test is more successful in detecting JB traffic. Where JB makes changes to most of the legitimate traffic IPDs that it uses, this is not the case for these other covert channels. TR-CTC does not alter the original traffic, but merely reorders it, while DPOI only alters the original traffic when a one is to be transmitted, so a decent portion of this traffic keeps the same shape in distribution (although lowered in occurrence). The covert channels based on models generate their own IPDs, which (for a good model) are difficult to distinguish from legitimate traffic in their distribution shapes. In Figure 4.22 these differences in shapes can be seen as the (truncated) cumulative distribution functions of the traffic without jitter, for legitimate, TR-CTC, and two variants of JB. Notable is that TR-CTC and the JB variant that uses the smallest w-value follow the training sample closely, while the larger variant deviates from these distributions by a large margin. Next to this, there are clear increments in the distributions for both JB variants. These increments are caused by the added jitter, which are rounded to millisecond precision, such that there is a step for each subsequent millisecond.

From the results we conclude that the KS-test is capable of detecting certain covert channel techniques, under specific conditions. For the simpler CTCs that create distinct IPDs, the detection scores are mainly dependent on the size of the values from their settings, as well as the amount of jitter present in the network. The KS-test is also able to detect JB traffic, but this is again dependent on the size of the w-values, and is reliant on the underlying traffic type that is used. In general this method is not suited for use on SSH traffic, despite having some decent true positive scores. The high false positive scores for SSH make it difficult to apply in a realistic situation.

4.9. Spearman-Rho, Wilcoxon Signed-Rank, Mann-Whitney-Wilcoxon

Last we examine the three non-parametric statistical tests of Spearman-Rho (SRHO), Wilcoxon Signed-Rank (WSR), and Mann-Whitney-Wilcoxon (MWW), discussed in Section 2.2.6. For these tests the overall application is the same, with the exception of what is being examined by each of them. From two subsequent samples of traffic from the same flow the ranks of each of the values within the sample are determined, according to the specific test. These ranks are then used to calculate the p-values for each of the different tests. Then, to examine how the traffic in the flow changes over time, the average over a set number of these p-values is calculated. This average is compared to a threshold value obtained from the same test performed on legitimate traffic, which then indicates whether covert traffic might be present. With regards to the underlying tests, SRHO measures the correlation between the ranks of the samples, while WSR, and MWW both test whether the samples are drawn from the same distribution. There are also some differences in how the ranking is handled for each of these tests. SRHO ranks the IPDs within each of the two samples independently according to their relative values, which are then used as pairs in the calculation of the p-value. In the case of WSR the ranks are determined by the absolute differences between each pair of values in the sample, resulting in one

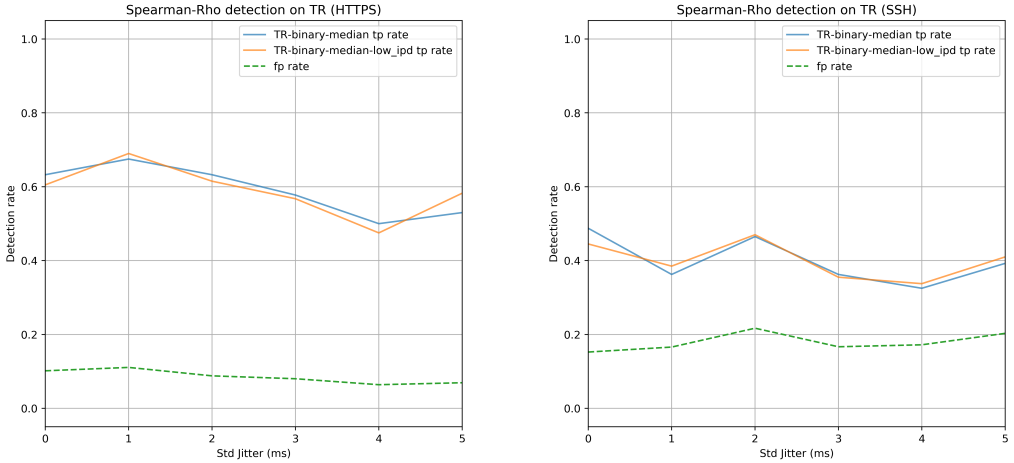


Figure 4.23: Spearman-Rho detection on TR-CTC

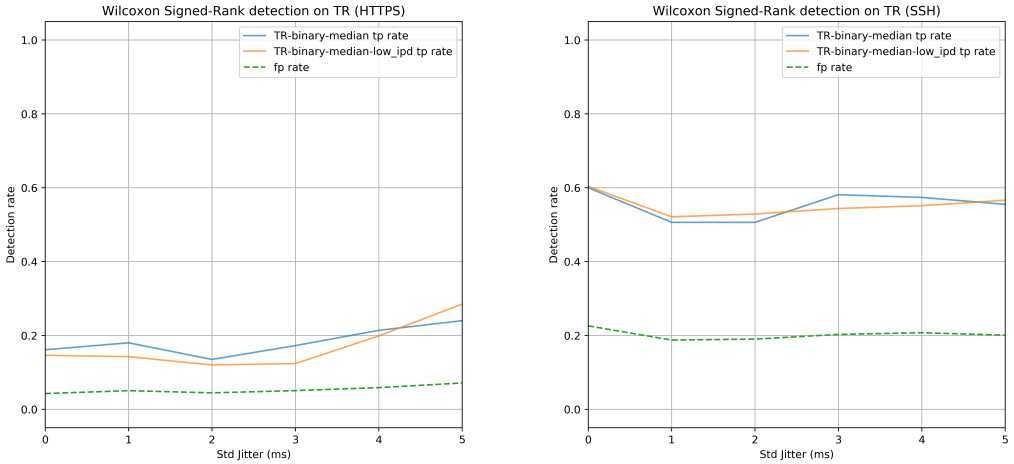


Figure 4.24: Wilcoxon Signed-Rank detection on TR-CTC

series of rankings for determining the p-value. MWW combines both samples and ranks all the values as a whole, that are then used as a single list of rankings in the calculation. The detection of CTCs by these methods is based on the premise that both legitimate and covert traffic will have grouped p-values that are distinct from each other.

One of the main advantages of these detection methods, due to being derived from non-parametric tests, is the usage of the relative positions within the samples. This means that the detection results are more determined by the ordering of the traffic and less so by the actual IPD values. Therefore these methods may detect the underlying process of CTCs such as MB-CTC or TR-CTC, where the network traffic is determined by a model or is based on legitimate traffic, which will produce similar (or the same) values as that traffic. Next to this, changes in the IPDs due to jitter or used variants will likely have little effect on the performance of these methods, since the overall ordering will be similar in these cases.

For SRHO performed on HTTPS traffic we report decent true positive rates, with an average of around 50 percent, for all covert channel techniques and their variants, under all amounts of jitter. In the case of SSH traffic, however, the detection rates for SRHO are notably lower on all covert channels. On the other hand, for both WSR and MWW we find the reverse of SRHO, where there are significant scores on most CTCs for SSH traffic, but not for HTTPS. The exception to these results for WSR and

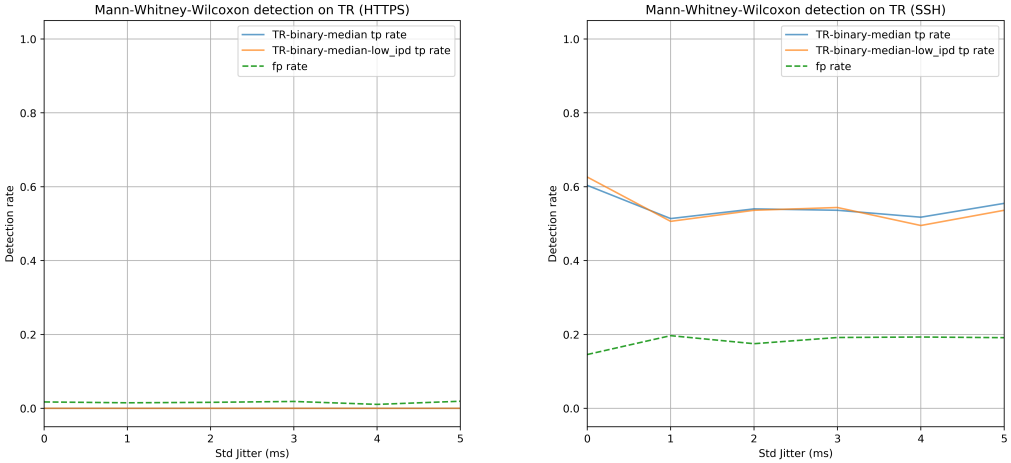


Figure 4.25: Mann-Whitney-Wilcoxon detection on TR-CTC

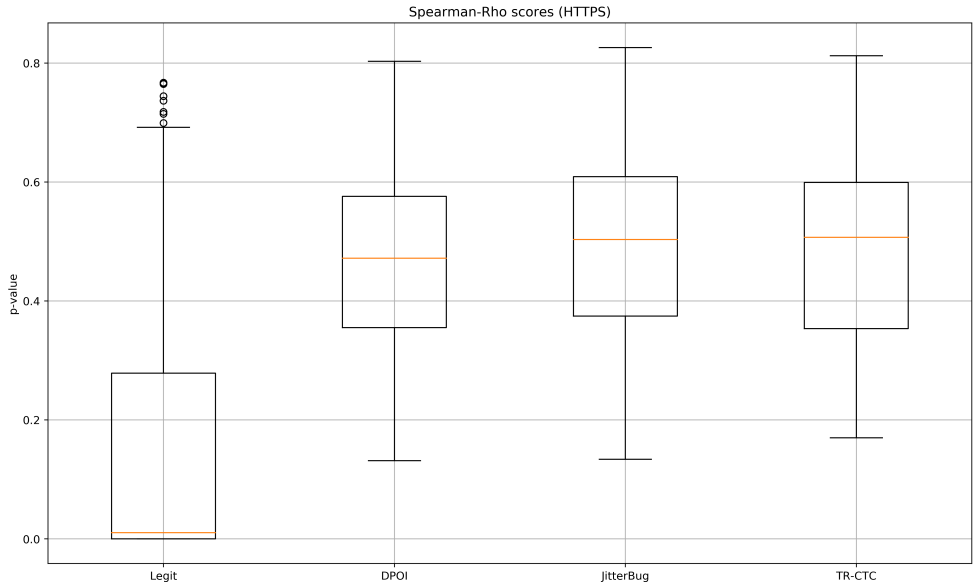


Figure 4.26: Spearman-Rho scores HTTPS

MWW are DPOI and JB, for which there are no significant true positive rates on either traffic type. Most notably we find that all three of these non-parametric methods have adequate detection of TR-CTC for either HTTPS or SSH traffic, which the previously discussed detection methods have not been able to achieve. In Figures 4.23, 4.24, and 4.25 the detection rates for TR-CTC are shown, using SRHO, WSR, and MWW respectively. From these figures we can see the overall performance on TR-CTC by all three methods, as well as the difference between both traffic types.

In the original work of these detection methods, using the same settings, the results were reported as between 98 and 100 percent true positive rates, and false positive rates between 0 and 6.66 percent. As we have shown in the results, this is distinctly higher than what we have obtained in our tests. We argue that these large differences in the observed performance are caused by some main concerns with how their methodology is set up and applied. First, we find that their overall attacker scenario, with

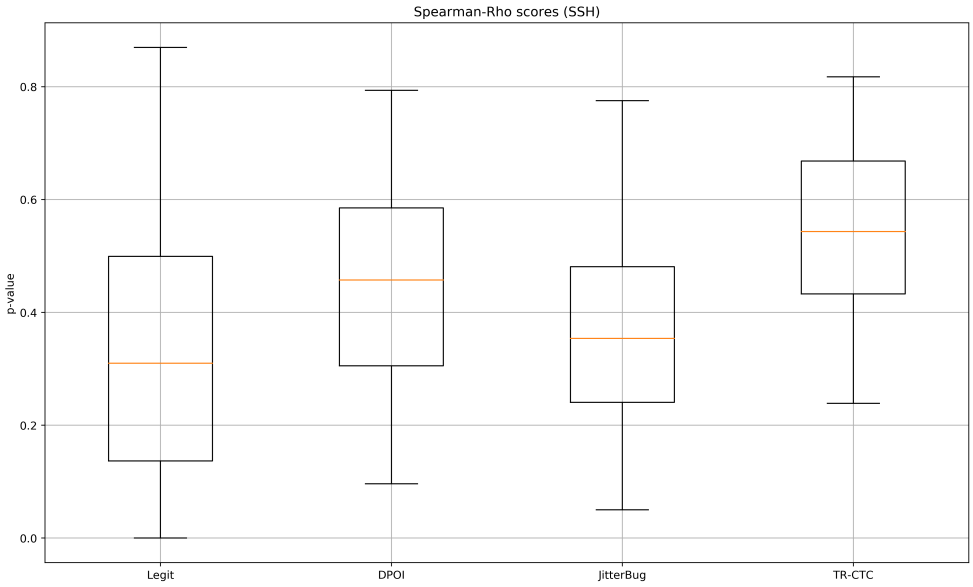


Figure 4.27: Spearman-Rho scores SSH

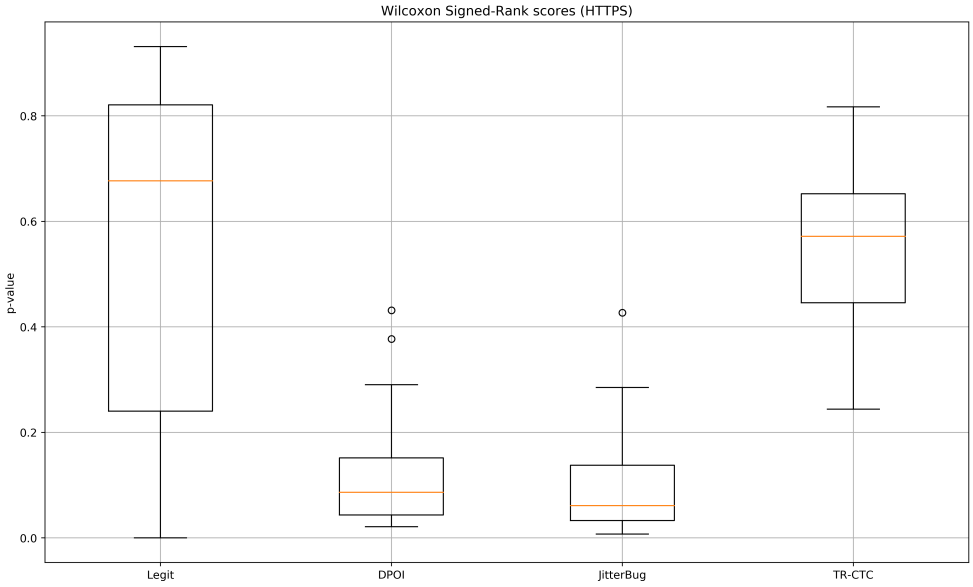


Figure 4.28: Wilcoxon Signed-Rank scores HTTPS

how covert traffic is transmitted, differs from ours. We have modeled this as the attacker using a single flow, over which only covert traffic is transmitted. In their case covert and overt traffic is intermittently transmitted over a single flow, more as a man-in-the-middle scenario. In this scenario the process has access to a legitimate flow, which it either passes through undisturbed or periodically alters to transmit its covert data. Although it might not necessarily be a wrong assumption that this could happen, it is

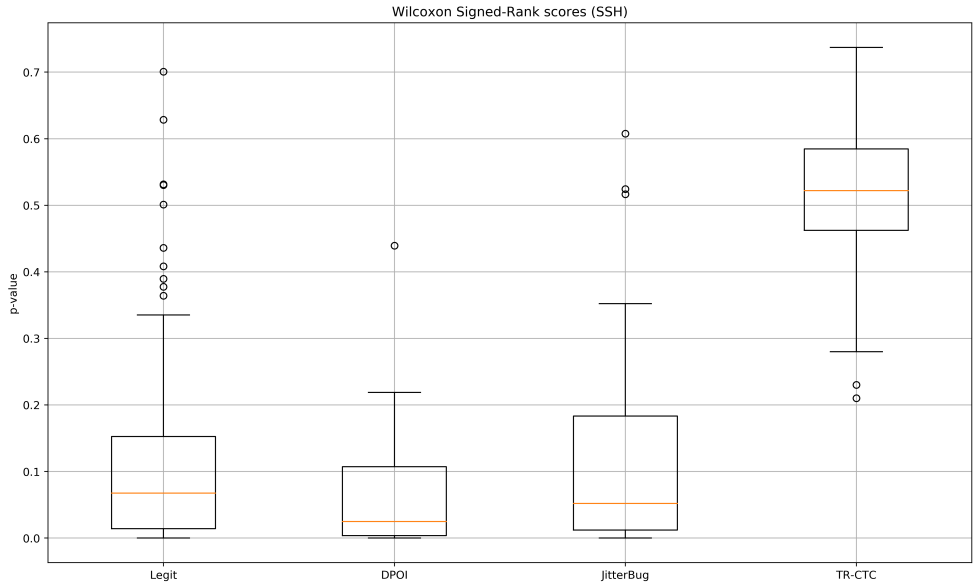


Figure 4.29: Wilcoxon Signed-Rank scores SSH

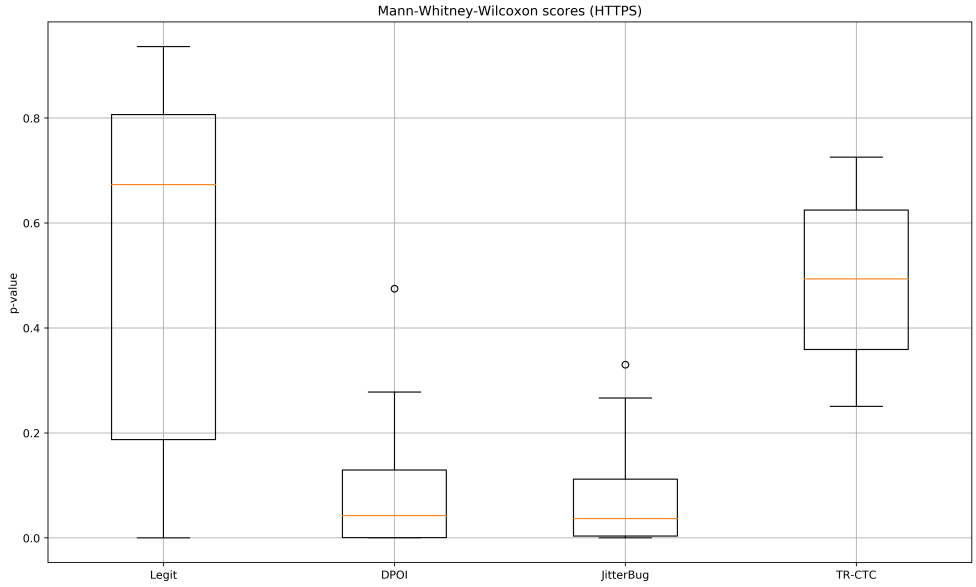


Figure 4.30: Mann-Whitney-Wilcoxon scores HTTPS

a somewhat unrealistic scenario to have multiple forms of covert traffic and different types of modeled legitimate traffic in the same flow. Second, and likely the largest concern, is what type of data is used to represent the legitimate traffic in their scenario. Next to the majority of traffic that is generated from different statistical models, there are only two flows of HTTPS traffic, recorded from two users. With the representative legitimate traffic either being from models or from a small sample of real traffic, it is

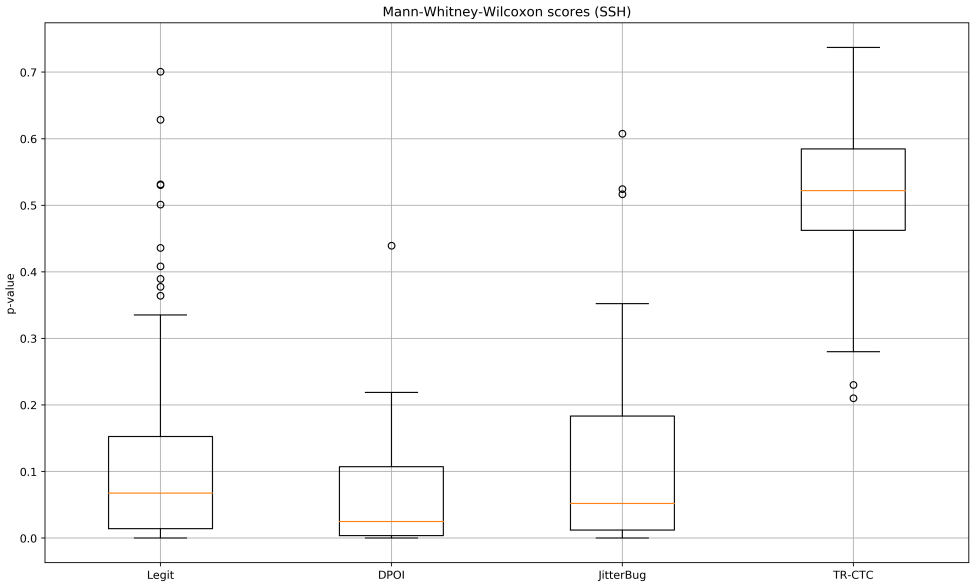


Figure 4.31: Mann-Whitney-Wilcoxon scores SSH

likely that they will be somewhat homogeneous. Without significant variation in the legitimate traffic, the difference between CTC and legitimate traffic might be more distinct, thus resulting in higher detection rates. As we will later show, there can be quite some variation in the legitimate traffic scores for these methods, when performed on a larger amount and different types of real traffic. Third, is how the thresholds are determined during the training phase and how they relate to the testing phase. By performing the detection methods on a training set of the data, an optimal threshold is chosen from the observed values for legitimate and covert traffic. This method could be a valid strategy of determining the threshold, if we can be certain that the same results are obtained for a wide range of network conditions and covert channel variants. Otherwise this method and the threshold values obtained from them are not generally applicable in all scenarios. As we have expressed in Section 3, there is also some concern with basing the thresholds off of values of certain covert channel techniques and variants. When it is not known beforehand which covert channels will be used by an attacker, determining the detection thresholds from other CTC techniques or variants could diminish detection performance in a general setting. Furthermore, if our point regarding homogeneity of legitimate traffic above holds, then results for training will be too similar to testing and the detection results might not be representative of a realistic situation. This same argument applies for the covert channel techniques that use the same underlying traffic streams for both training and testing, as well as having little variation in which settings are used for them.

Aside from the concerns that stem from the original work, there are some general performance differences that we have observed in our tests. To give insight into why the differences between the two traffic types and the detection methods occur, we provide the scores for each of the detection methods. For this purpose we have chosen to compare the legitimate traffic scores against those for TR-CTC, DPOI, and JB. The scores can be seen, for HTTPS and SSH respectively, in Figures 4.26 and 4.27 for SRHO, in Figures 4.28 and 4.29 for WSR, and in Figures 4.30 and 4.31 for MWW. From these figures we can see that the overall scores for these shown CTCs are consistent for both traffic types. However, the main difference between the detection methods lies in the scores for the legitimate traffic. For SRHO we can observe scores more close to zero for legitimate HTTPS traffic, while they are higher for SSH traffic. On the other hand, for both WSR and MWW we note the legitimate scores more grouped around zero for SSH, with the scores higher and more spread out for HTTPS traffic. So in the cases where there is significant overlap for the legitimate traffic scores and the CTCs, it becomes

difficult to sufficiently distinguish between overt and covert traffic. This can be clearly seen for WSR and MWW with the HTTPS traffic in Figures 4.28 and 4.30, where TR-CTC is fully overlapped by the majority of the values of the legitimate traffic. Another interesting observation to note are the scores for DPOI and JB for WSR and MWW, specifically for the HTTPS traffic type. These scores are significantly lower than the legitimate traffic for this traffic type, which could then be correctly classified by these detection methods. However, these detection methods are applied using an upper threshold, since the original work specifies that an upper threshold should be used for all three detection methods. This means that the scores for these two CTCs that are under the legitimate traffic cannot be correctly classified. These scores indicate that it might thus be beneficial to use either a two-sided threshold or a lower threshold direction, for these specific cases of covert channels and traffic type.

4.10. Overview

To give insight into the overall performance of the detection methods we give overviews of the full results of the analyses. For each combination of detection method and covert channel technique we provide an aggregated rating, over all variants and amounts of jitter. We give this rating on the basis of a simple scoring system from the false and true positive rates that are seen in the results. At each point of jitter a positive score is noted if the true positive rate is 50 percent or higher, with the false positive rate no higher than 20 percent. If a combination of detection method and covert channel only has positive scores, the rating is shown as positive as well, indicated by a plus (+). Conversely, when either the false positive rate is over 20 percent or the true positive rate is under 50 percent a negative score is given. A negative rating is then shown, given by a minus (-), if there are only negative scores in the combination. If there are both positive and negative scores occurring within the same combination, we mark this as a mixed result, given by a tilde (∼). The overview of the ratings for HTTPS are given in Table 4.1, and for SSH in Table 4.2.

In an ideal situation we would require zero false positives and 100 percent true positive rates for a positively rated detection method. However, since these scores are unlikely to be attainable, we set conditions under which the detection methods might need to perform. These conditions mostly depend on the requirements of the environments in which these detection methods will be applied. For example, in an environment where there is a small amount of overt traffic the false positive rate could be of less significance, compared to one where there is more traffic. In the case of more traffic, the higher amount of false positives would require more resources to further investigate the increase in possible covert channels, that are indicated by the detection methods. Depending on the importance of detecting covert channel traffic in the environment, even detection methods with lower true positive rates may be of some use, when they have the possibility to detect some traffic where others do not. Since there is no precedent as to what the required conditions are in a real scenario, we provide a somewhat arbitrary estimate of what they could be. The ratings do not provide a full conclusion on the applicability of the detection methods, but it does allow for a comparison between detection methods and provides a simplified overview of our complete results.

Next to the overall performance of each of the detection methods, we also indicate in the tables which method is likely the most suitable singular solution for each of the CTCs. This decision is primarily determined by the true positive and false positive rates we have obtained, with a preference given to the more consistent detection methods. When comparing two detection methods, one might have higher overall scores, but some low scores for one or more specific variants, while the other has consistent (but lower) scores for all variants. The most suitable solution is then chosen to be the method that has these consistent scores for all variants. An example of this can be seen for DPOI, using HTTPS traffic, for the KLD and SRHO detection methods. For this particular CTC, KLD has higher scores than SRHO for most of the variants, with the exception of one where it does not perform well. Therefore, the preference is given to SRHO, which scores consistently well over all variants of DPOI. The main reason for this decision is that in the case that an attacker can find out which detection methods are used, they could choose a CTC variant that is less likely to be detected by these methods. To be able to defend against such a scenario it is thus beneficial to use a metric that performs consistently well, even though some of its scores might be lower overall. It should be noted that it is possible to use both of these described metrics simultaneously, to cover more of a certain CTC's traffic, at the cost of more computational power and amount of results to verify.

	SCC	On-Off	DPOI	JB	MB-CTC	RMB-CTC	TR-CTC	LBtNP
Regularity	+	-	-	-	-	-	-	~
ε-similarity	+	+	-	-	~	~	-	~
Compressibility	+	+	~	~	-	-	-	~
Mean IPD	-	-	-	-	-	-	-	-
KS test	~	~	-	~	-	-	-	~
EN	~	~	-	-	-	-	-	-
CCE	-	-	-	-	~	~	-	~
KLD	+	+	~	~	-	-	-	~
Welch's t-test	-	-	-	-	~	~	-	~
Kurtosis mean	+	+	-	-	-	-	-	~
Kurtosis std	+	-	-	-	-	-	-	~
Kurtosis regularity	~	-	-	-	-	-	-	~
Skew mean	+	+	-	-	-	-	-	~
Skew std	+	-	-	-	-	-	-	~
Skew regularity	-	-	-	-	-	-	-	-
SRHO	+	~	~	~	~	~	~	~
WSR	~	-	-	-	-	-	-	-
MWW	-	-	-	-	-	-	-	-

Table 4.1: Detection performance overview HTTPS

	SCC	On-Off	DPOI	JB	MB-CTC	RMB-CTC	TR-CTC	LBtNP
Regularity	+	~	-	-	-	-	-	~
ε-similarity	~	~	-	-	-	-	-	~
Compressibility	+	~	-	-	-	-	-	~
Mean IPD	-	-	-	-	-	-	-	-
KS test	~	~	-	-	-	-	-	-
EN	~	~	-	-	-	-	-	-
CCE	-	~	+	+	~	~	-	~
KLD	~	~	~	~	~	~	-	~
Welch's t-test	-	-	-	-	-	-	-	-
Kurtosis mean	+	-	-	-	-	-	-	~
Kurtosis std	+	-	-	-	-	-	-	~
Kurtosis regularity	~	-	-	-	-	-	-	~
Skew mean	~	-	-	-	-	-	-	-
Skew std	+	-	-	-	-	-	-	~
Skew regularity	-	-	-	-	-	-	-	-
SRHO	-	-	-	-	-	-	-	-
WSR	~	~	-	-	~	-	~	~
MWW	~	~	-	-	+	~	+	~

Table 4.2: Detection performance overview SSH

5

Conclusion

In this thesis we have performed a broad performance analysis for the current detection techniques on the most prevalent covert timing channels. This analysis was performed using real network traffic from the TU Delft, under varying network conditions, in the form of simulated jitter. From this we have largely seen that the performance described in previous work might not be as attainable in a more realistic setting. We have shown that the difference in performance between our work and that of previous work can be caused by a number of different factors. The first main factor that we identified is what kind of traffic is used in the analyses in previous work. These analyses are largely performed on modeled traffic, that is presumed to represent a real scenario, with in some cases actual recorded traffic being used. In the case of real traffic we have found that there is either too little of this being used, or the data could possibly be outdated and not represent current networks. We have thus argued that this traffic is possibly not representative of legitimate traffic in current, real network settings. This can then be a reason why we have observed differences in performance for these types of traffic and our traffic. Second, we found that multiple works did not account for the effect of network jitter in their analyses. Any network contains at least some form of jitter, so in a comparison to real network scenarios jitter should be taken into consideration. This network effect, and its consequence to the resulting detection, especially becomes clear in the case of CTCs that use a limited amount of set values. For these CTCs their observed IPDs are concentrated around these sets of values without jitter, while there is more variation when an amount of jitter is added. This has then shown considerable differences in results in our work, compared to the previous work that did not include jitter. Third, we found that there were some assumptions that were made, without significant evidence to back them up. One of the assumptions that lead to a low performance can be seen for the mean IPD method, in Section 4.2. In this case, the legitimate traffic is presumed to be normally distributed, which would theoretically lead to correct classification for certain covert channel techniques. As we have shown, this was not how the traffic was distributed in our real traffic. Next to this, previous work has performed their analyses on a small amount of CTC variants, which makes assumptions on what an attacker would choose. If the attacker has knowledge of which detection method is being performed, they could exploit the weakness of this detection method by choosing CTC values that provide lower detection rates. By using a wide range of possible CTC values, we assume less on what an attacker would choose, and this thesis has shown how the usage of different covert channel variants has an effect on the detection methods. Last, we found large differences in detection performance for the two different traffic types of HTTPS and SSH, for many of the detection methods. This indicates that a high detection rate for one type of traffic cannot be directly translated into good results for another. The results obtained for one traffic type, as is mostly done in previous work, thus does not guarantee any significant results for other application traffic. Due to these factors we argue that the results obtained in previous work are likely not indicative of their actual performance, when applied in a real network scenario. From the performed analyses we provide answers to the research questions posed in Section 1:

> *Research question 1: Is there a single detection method or combination of detection methods that can sufficiently detect the most prevalent covert timing channels?*

To answer this research question we mainly look at the devised rating scheme for the detection methods, discussed in Section 4.10. From the overviews of the ratings for HTTPS and SSH, provided in Tables 4.1 and 4.2, we can observe that the detection of these covert timing channels is likely not sufficient in its current state. Under this rating system, this would be indicated by at least one fully positive rating in every column representing the covert channels, which is not present in either of the tables. However, we did find that there are some CTC techniques that can be sufficiently detected, using particular detection methods. For both traffic types the SCC, due to its simplicity, has multiple detection methods that are able to achieve adequately high detection rates. The same goes for the On-Off covert channel, but only for HTTPS the detection scores are consistently high enough to attain positive ratings. Next to this, we found positive ratings for SSH traffic with the use of CCE on DPOI and JB, as well as for MWW on MB-CTC and TR-CTC. From the overview presented in the tables we can also observe that the (model variant of) LBtNP, and RMB-CTC are two of the more difficult CTC techniques to detect. This is due to these covert channels having no positive scores for either traffic type, under the discussed rating scheme. Further, we find that some detection methods, with improvements to the performance, could be used as a means for general detection of the covered covert channels. In the case of HTTPS traffic, SRHO shows decent detection over all CTC techniques, and even though these are mixed results, most detection scores are close to the true positive threshold we have set. Likewise, MWW achieves similar results for most covert channels using SSH traffic, with the exception of DPOI and JB, for which it has negative ratings. However, the gap in performance for these two covert channels techniques can be supplemented by the use of CCE, which does well in these particular cases. Thus, although their performance is currently not at a high enough standard, these three detection methods could in the future be used to detect a wide range of CTC techniques.

Research question 2: How does the existence of network effects, such as jitter, affect the performance of detection methods?

From performing the performance evaluation on the detection methods using varying amounts of network jitter, we were able to show the effects that jitter has on detection. We have seen that jitter has varying effects on the performance of detection methods, depending on the overall method with which each detection method calculates their scores and which covert channel is being detected. The most notable case for the negative effects of jitter can be seen for the method based on the first order entropy, in Section 4.1. For this particular detection method the addition of any amount of jitter causes the true positive rates of the method to decrease to zero for all covert channels. This then means that this method cannot be used in a realistic situation, provided that there is likely some amount of jitter present in each network. The effects of jitter for the detection of the different covert channels mainly depends on the CTC technique, and the relative size of the values that are used, compared to the amount of added jitter. For example, without jitter SCC might have a clear distinction from legitimate traffic, while the calculation for some detection methods provide similar results for both sets of traffic with the addition of jitter. In comparison, methods that model legitimate traffic and those that apply their own random values are more likely to be similar to legitimate traffic with or without jitter. Next to this, we found an interesting case for the compressibility detection method, discussed in Section 4.5. For this detection method there is actually an increase in detection performance for certain covert channels, between no jitter and a small amount of added jitter. Further, we have observed instances where detection methods have shown robustness against the effects of jitter. This is the case for all three detection methods based on non-parametric tests, in Section 4.9, as well as for CCE, discussed in Section 4.1. Why jitter has little effect on the performance is caused by two main factors in the operation of these detection methods. For the non-parametric test this is because they are less reliant on the actual IPD values, but more on the ordering, which is similar with or without jitter. On the other hand, for CCE this is mainly because of the large bins that it employs, which then places values in roughly the same bin regardless of jitter. In realistic scenarios, where there could be a varied amount of jitter present in the network, robustness against jitter is a desirable trait. This further adds to the point mentioned in answering the previous research question, that the three named detection methods of SRHO, MWW, and CCE might be more suitable to be applied in these network scenarios.

6

Future work

We believe that this thesis has provided valuable insight into the current state of covert timing channel detection, however there are some limitations in what our work can show. In this section we thus identify multiple directions where future research could be performed on this subject.

Directly following from the analyses and the comparison of our results to those brought forth in previous work, we find that there is a need for validation of these results, under realistic network conditions. We have seen that much of the previous work that has been done, performs their analyses on traffic that is likely to not be representative of legitimate traffic. The conditions for realistic networks should at the least involve the use of real network traffic, and an addition of some amount of (simulated) network jitter. Even the legitimate traffic used for our evaluation is limited to that of one location, so this might not represent traffic in a general network scenario. So our work should also be reproduced with traffic from other networks, to lend more credibility to the results that we have obtained. Next to this, since our work only covers two different types of application traffic, future work could also focus on other types of traffic and their effects on detection performance. We have already seen that there are major differences between these two traffic types, however there is still little information on the detection for other traffic types.

From our results we have discussed some indications that several detection methods could provide better performance, with adjustments in their operation. Therefore, there is possible advancements to be made by improving on existing detection methods, to better distinguish between overt and covert traffic. Next to this, future work should continue to introduce new detection methods, possibly related to those that we have shown to perform decently well. One of the existing detection methods, that we did not cover in this work, is the machine learning based detection of SVM. This is due to the scale of what was needed to effectively test the performance of SVM, which would require the use of different (amounts of) combinations of the other discussed detection methods. So a direction for future research is possibly in testing the detection capabilities of this particular method, with a wide range of combinations. A starting point for this research could be to apply this with the detection methods, that we have found to provide decent detection as a single method.

We have also identified that there are some possible improvements to be made to the overall methodology in our work. One such improvement could take the form of a pre-filtering step, for both training and the actual detection. This filtering should be able to remove part of the legitimate outliers within the traffic, and thus reduce the relatively high false positive rates we have seen. Next to this, there could also be a need for a system to determine the settings for the detection methods, depending on the type of traffic present. We have run our experiments with detection methods only using one setting, under the assumption that performance should be similar to that of previous work, if they are to be generally applied. However, there might be different settings for each of the detection methods, that provide a more optimal performance for the different traffic types. The methodology we have utilized in our experiments provides classification for samples within flows, which results in general detection rates of all traffic. It could perhaps be a better approach to label entire flows instead, by noting how often samples of such a flow are classified as either covert or overt traffic. In the case of flows consisting purely of covert traffic, this could occur more regularly than for legitimate traffic. On the other hand, sparse indications of covert traffic within a flow could just be outliers for legitimate traffic, which adds

up to the overall false positive rates.

Another direction, which is less related to the subject of detection, is the possible interference to covert communication caused by the added jitter to the network traffic. In this work we have seen that there is some significant effect that the jitter has on the detection methods, however we have not closely examined what its effects are on the covert channels. Some future research could thus be performed on this form of communication disruption for covert timing channels. It could be interesting to examine at which point jitter causes covert communication to be ineffective, due to a certain amount of bit errors caused by the network jitter. With the interference it should also be taken into account how much it affects legitimate communication of users on the network, in relation to what the network is used for and the importance of disrupting covert communications. This method of disruption could then possibly be used in combination with some of the detection methods, that we have shown to be more robust against the effects of jitter.

Bibliography

[1] Rennie Archibald and Dipak Ghosal. A covert timing channel based on fountain codes. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 970–977. IEEE, 2012.

[2] Rennie Archibald and Dipak Ghosal. A comparative analysis of detection metrics for covert timing channels. *Computers & Security*, 45:284 – 292, 2014. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2014.03.007. URL http://www.sciencedirect.com/science/article/pii/S0167404814000467.

[3] Vincent H. Berk, Annarita Giani, and George Cybenko Thayer. Detection of covert channel encoding in network packet delays. 2005.

[4] Serdar Cabuk. *Network covert channels: Design, analysis, detection, and elimination*. PhD thesis, Purdue University, 2006.

[5] Serdar Cabuk, Carla E. Brodley, and Clay Shields. Ip covert timing channels: Design and detection. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, CCS '04, pages 178–187, New York, NY, USA, 2004. ACM. ISBN 1-58113-961-6. doi: 10.1145/1030083.1030108. URL http://doi.acm.org/10.1145/1030083.1030108.

[6] Serdar Cabuk, Carla E. Brodley, and Clay Shields. Ip covert channel detection. *ACM Trans. Inf. Syst. Secur.*, 12(4):22:1–22:29, April 2009. ISSN 1094-9224. doi: 10.1145/1513601.1513604. URL http://doi.acm.org/10.1145/1513601.1513604.

[7] Steven Gianvecchio and Haining Wang. Detecting covert timing channels: An entropy-based approach. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 307–316, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-703-2. doi: 10.1145/1315245.1315284. URL http://doi.acm.org/10.1145/1315245.1315284.

[8] Steven Gianvecchio and Haining Wang. An entropy-based approach to detecting covert timing channels. *IEEE Transactions on Dependable and Secure Computing*, 8(6):785–797, 2010.

[9] Steven Gianvecchio, Haining Wang, Duminda Wijesekera, and Sushil Jajodia. Model-based covert timing channels: Automated modeling and evasion. In *International Workshop on Recent Advances in Intrusion Detection*, pages 211–230. Springer, 2008.

[10] Butler Lampson. A note on the confinement problem. 1973.

[11] Ira S Moskowitz and Allen R Miller. Simple timing channels. In *Proceedings of 1994 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 56–64. IEEE, 1994.

[12] Sheng Mou, Zhiwen Zhao, Sisi Jiang, Zushun Wu, and Jiaojiao Zhu. Feature extraction and classification algorithm for detecting complex covert timing channel. *Computers & Security*, 31 (1):70 – 82, 2012. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2011.11.001. URL http://www.sciencedirect.com/science/article/pii/S0167404811001349.

[13] Shishir Nagaraja, Amir Houmansadr, Pratch Piyawongwisal, Vijit Singh, Pragya Agarwal, and Nikita Borisov. Stegobot: a covert social network botnet. In *International Workshop on Information Hiding*, pages 299–313. Springer, 2011.

[14] Pai Peng, Peng Ning, and Douglas S Reeves. On the secrecy of timing-based active watermarking trace-back techniques. In *2006 IEEE Symposium on Security and Privacy (S&P'06)*, pages 15–pp. IEEE, 2006.

[15] F. Rezaei, M. Hempel, P. L. Shrestha, S. M. Rakshit, and H. Sharif. Detecting covert timing channels using non-parametric statistical approaches. In *2015 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 102–107, Aug 2015. doi: 10.1109/IWCMC.2015.7289065.

[16] F. Rezaei, M. Hempel, and H. Sharif. Towards a reliable detection of covert timing channels over real-time network traffic. *IEEE Transactions on Dependable and Secure Computing*, 14(3):249–264, May 2017. ISSN 1545-5971. doi: 10.1109/TDSC.2017.2656078.

[17] Fahimeh Rezaei, Michael Hempel, Pradhumna Lal Shrestha, and Hamid Sharif. Achieving robustness and capacity gains in covert timing channels. In *2014 IEEE International Conference on Communications (ICC)*, pages 969–974. IEEE, 2014.

[18] Gaurav Shah, Andres Molina, Matt Blaze, et al. Keyboards and covert channels. In *USENIX Security Symposium*, volume 15, 2006.

[19] Gustavus J Simmons. The prisoners' problem and the subliminal channel. In *Advances in Cryptology*, pages 51–67. Springer, 1984.

Appendix A

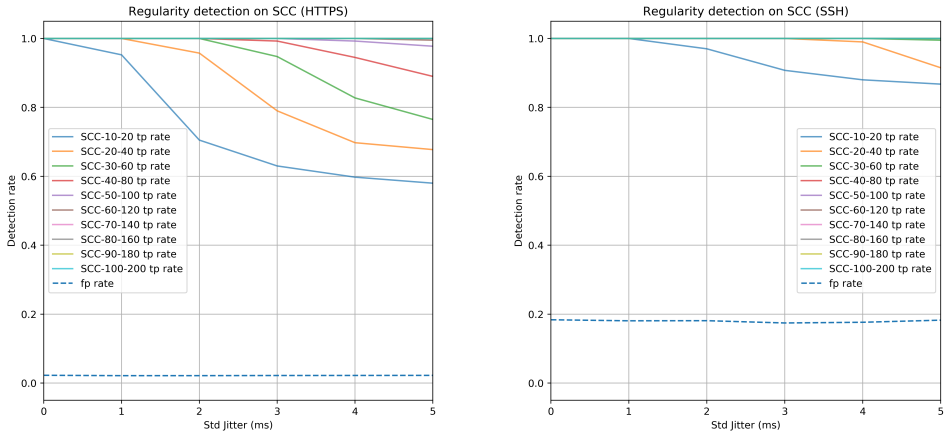


Figure 1: Regularity detection on SCC

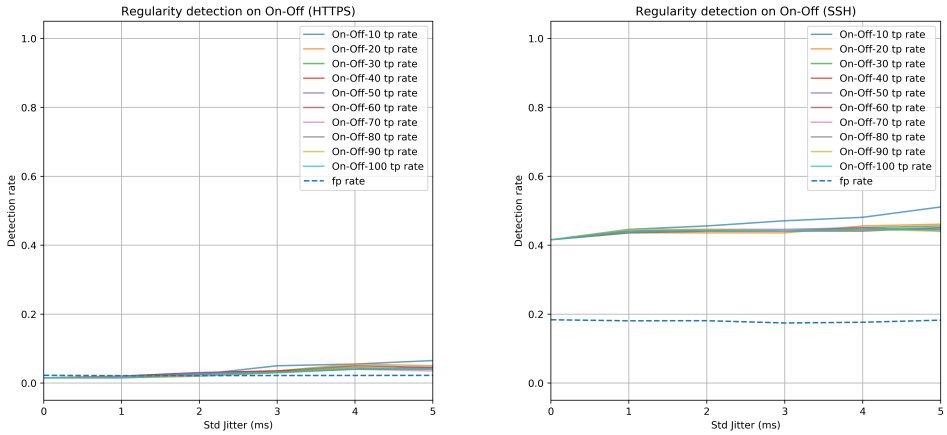


Figure 2: Regularity detection on On-Off

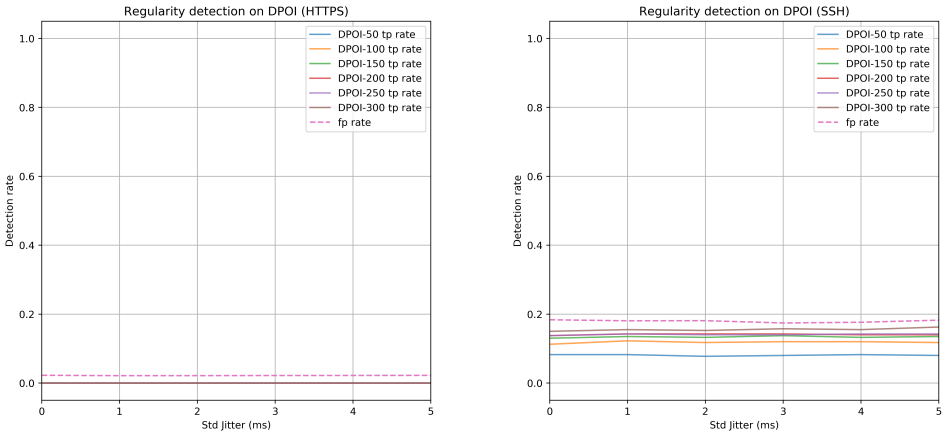


Figure 3: Regularity detection on DPOI

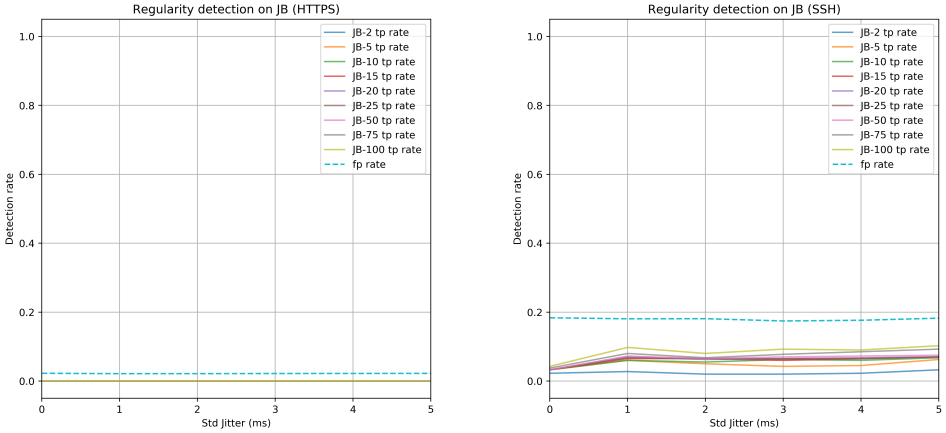


Figure 4: Regularity detection on JB

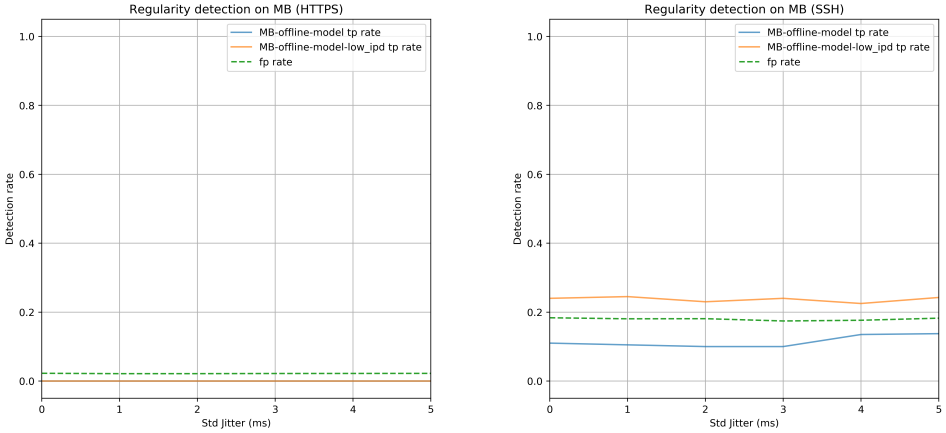


Figure 5: Regularity detection on MB-CTC

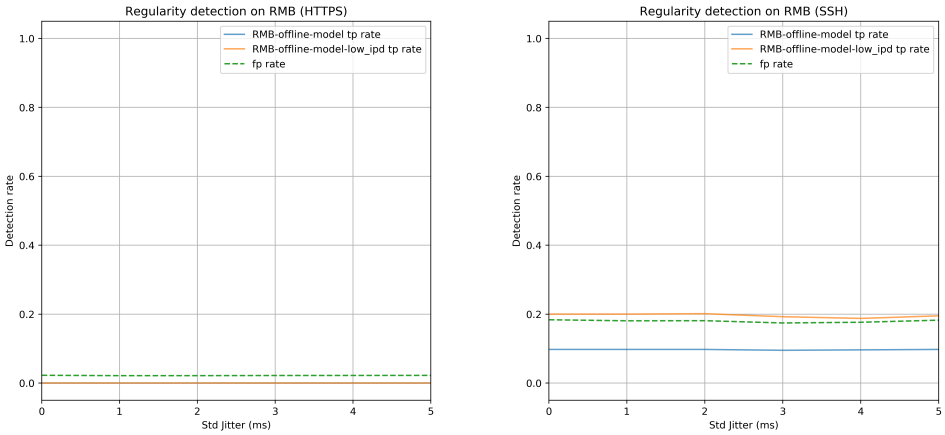


Figure 6: Regularity detection on RMB-CTC

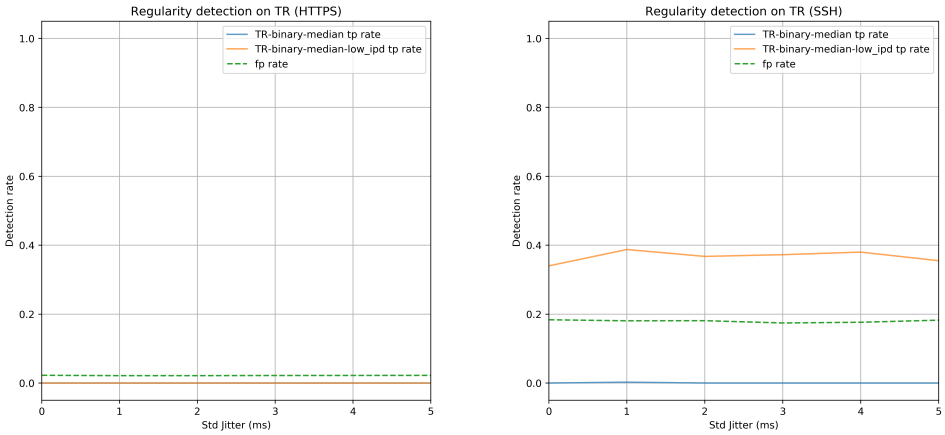


Figure 7: Regularity detection on TR-CTC

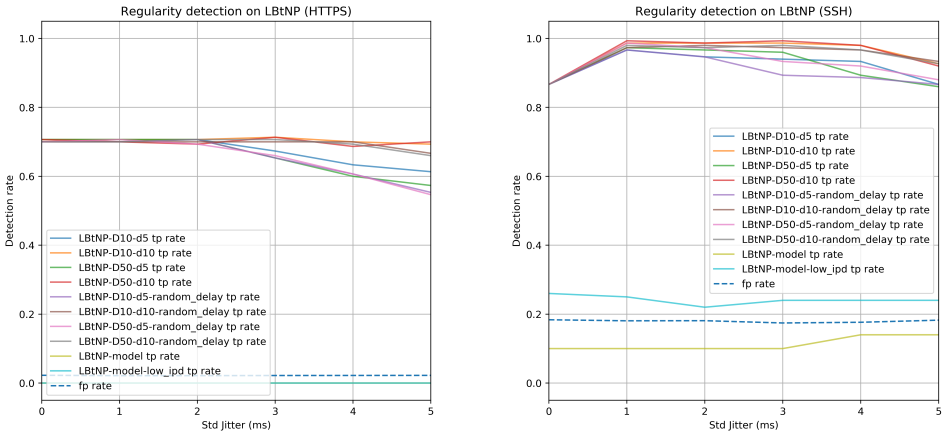


Figure 8: Regularity detection on LBtNP

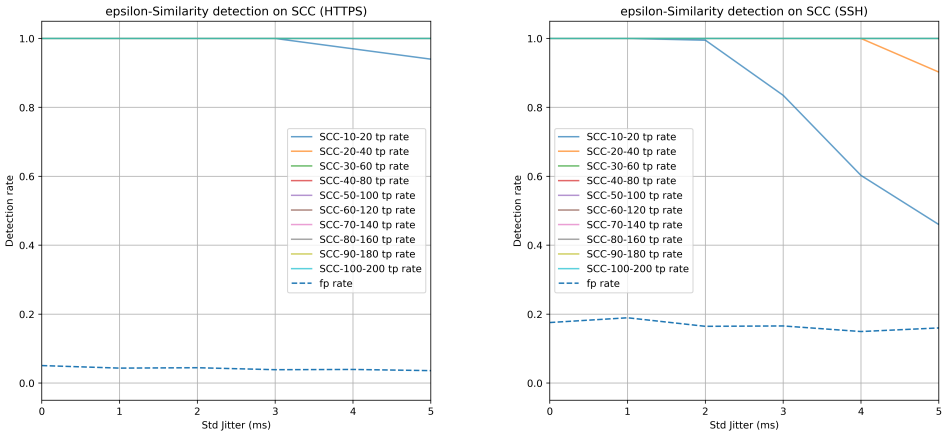


Figure 9: epsilon-Similarity detection on SCC

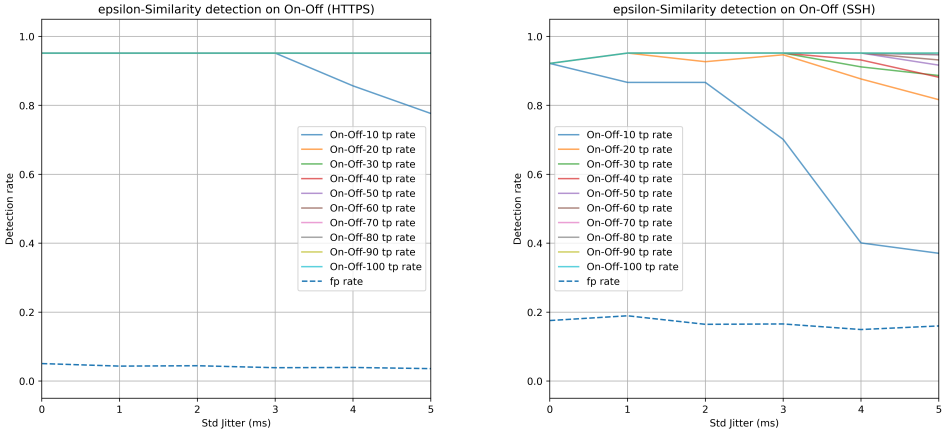


Figure 10: epsilon-Similarity detection on On-Off

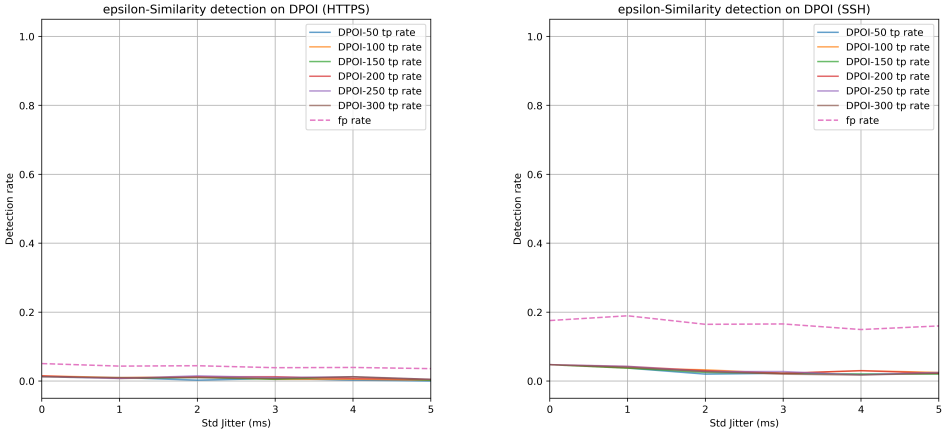


Figure 11: epsilon-Similarity detection on DPOI

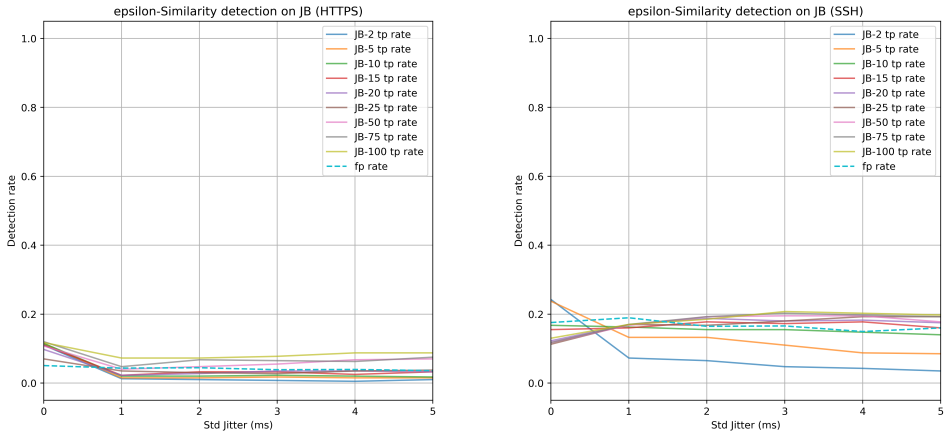


Figure 12: epsilon-Similarity detection on JB

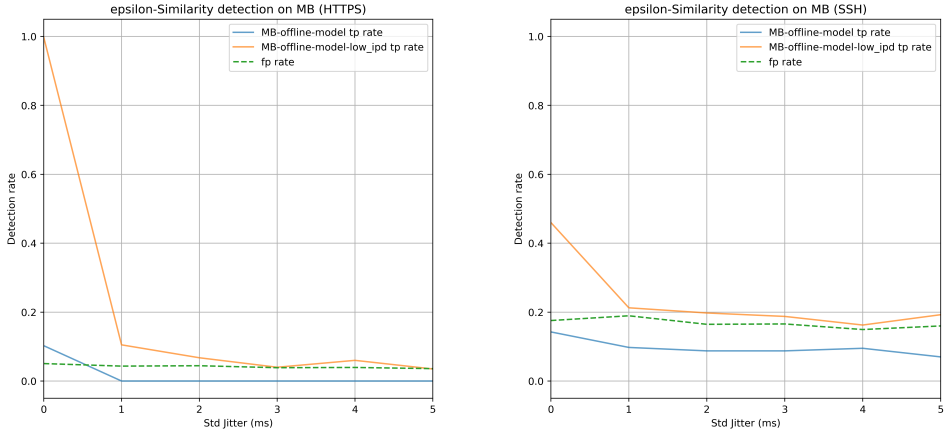


Figure 13: epsilon-Similarity detection on MB-CTC

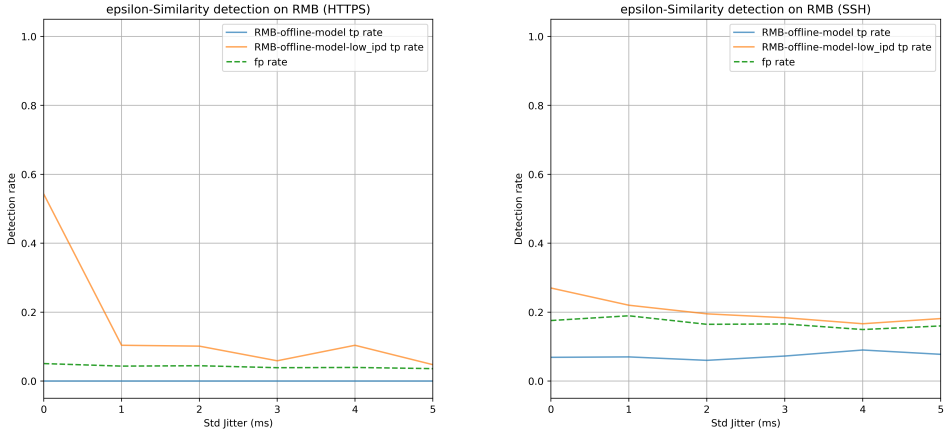


Figure 14: epsilon-Similarity detection on RMB-CTC

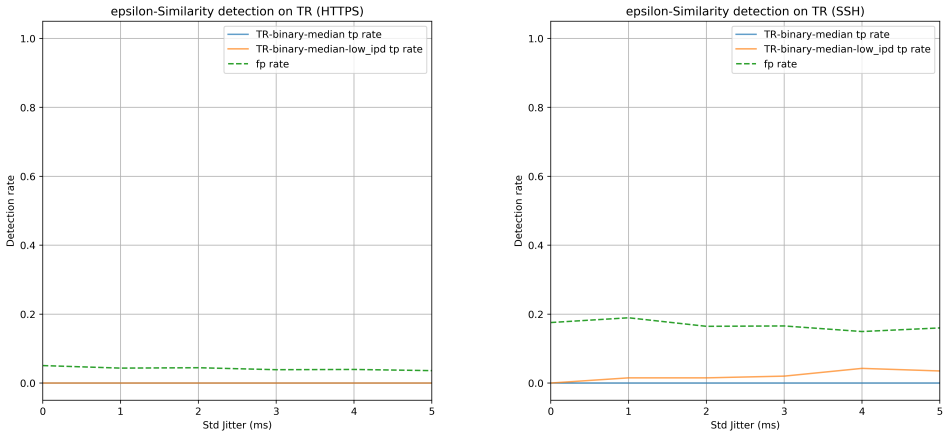


Figure 15: epsilon-Similarity detection on TR-CTC

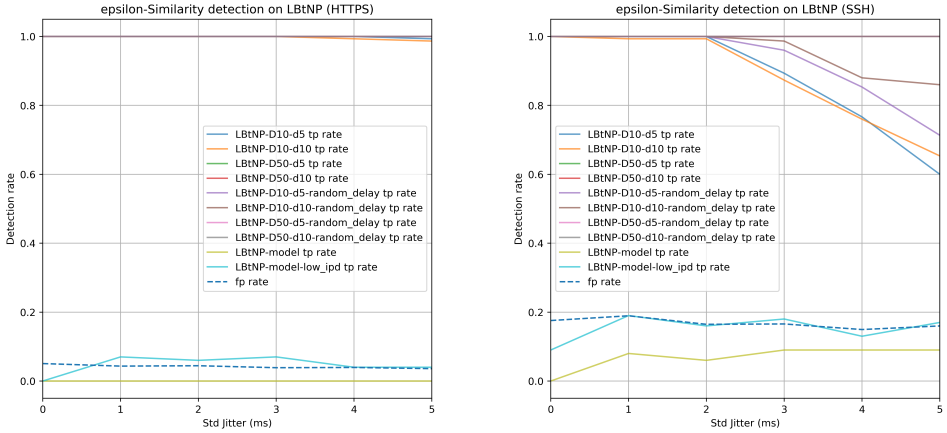


Figure 16: epsilon-Similarity detection on LBtNP

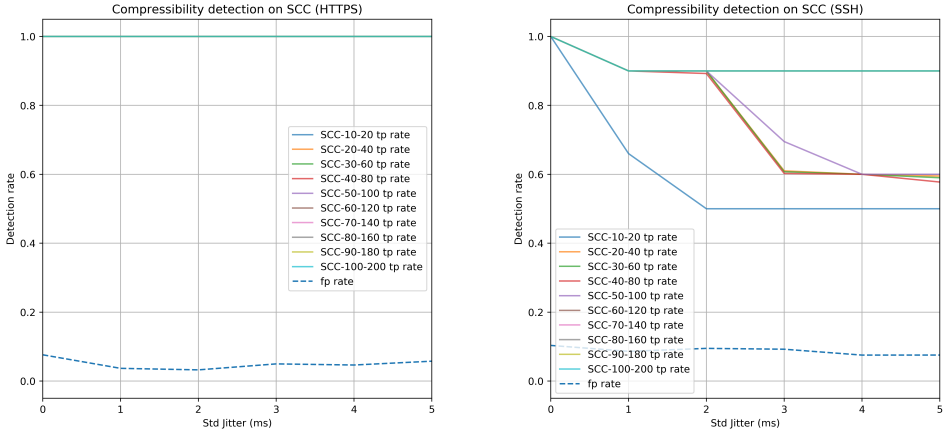


Figure 17: Compressibility detection on SCC

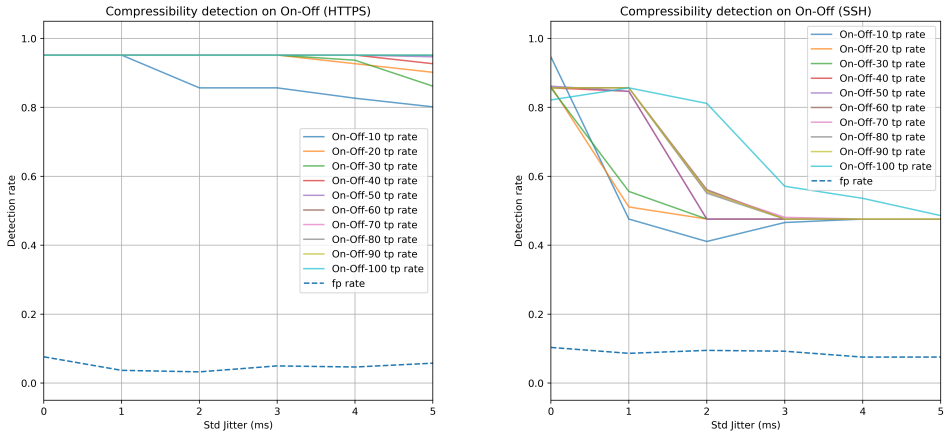


Figure 18: Compressibility detection on On-Off

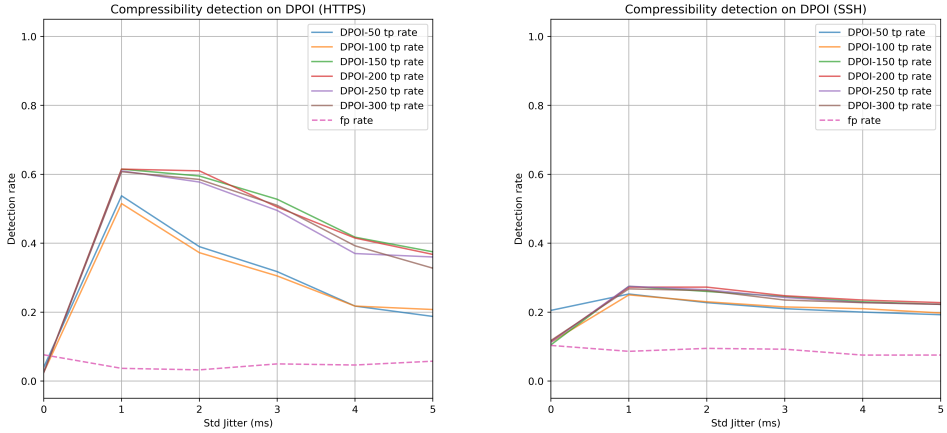


Figure 19: Compressibility detection on DPOI

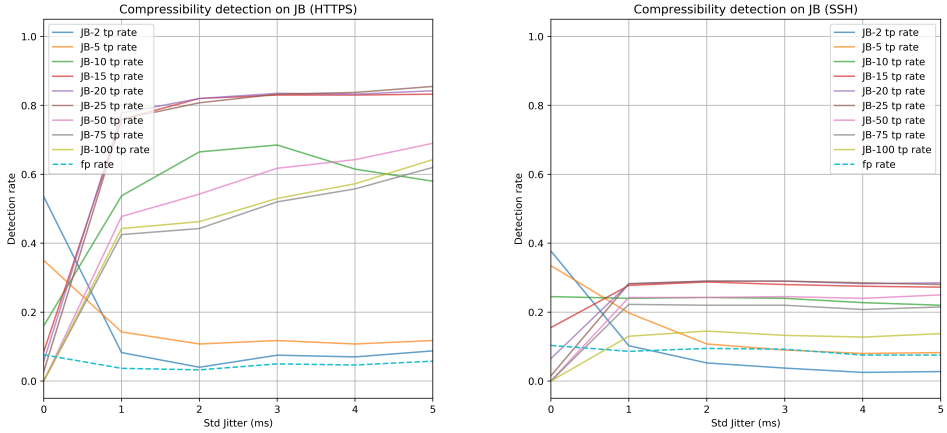


Figure 20: Compressibility detection on JB

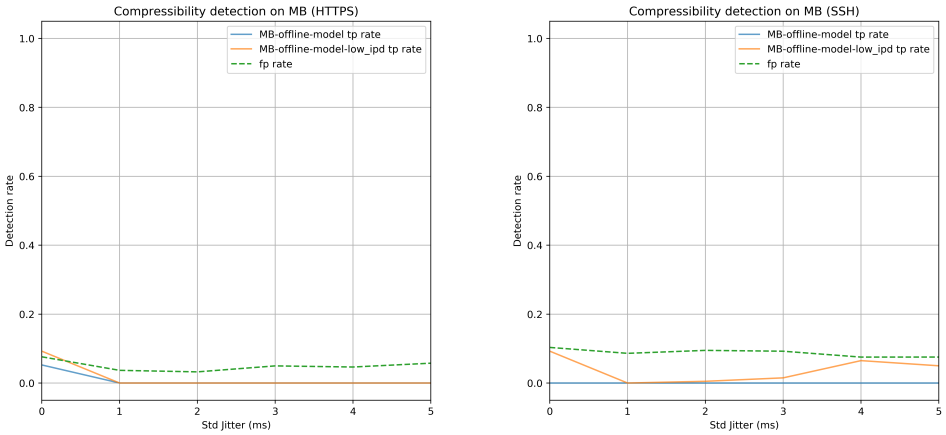


Figure 21: Compressibility detection on MB-CTC

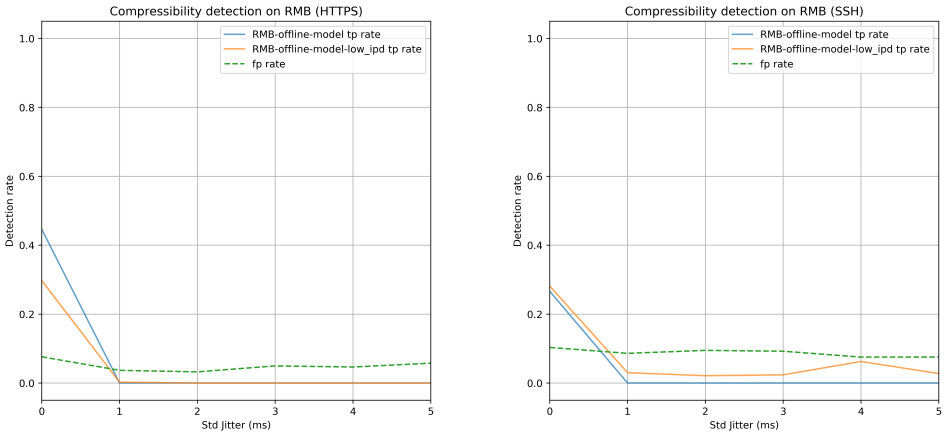


Figure 22: Compressibility detection on RMB-CTC

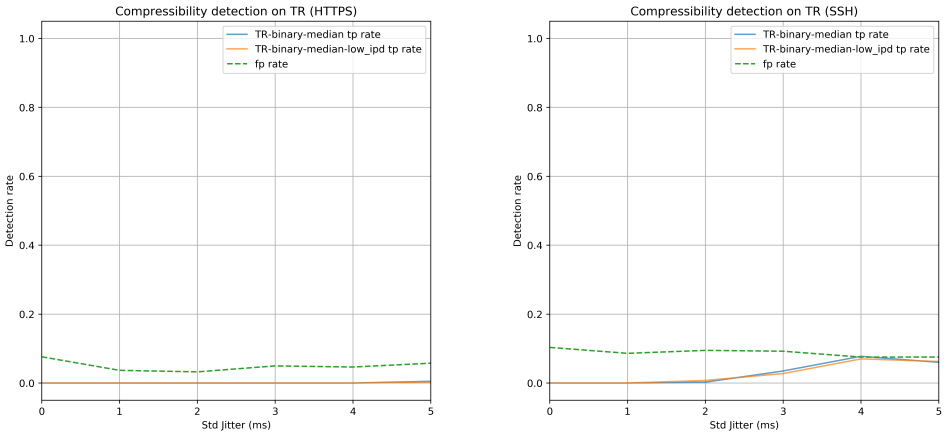


Figure 23: Compressibility detection on TR-CTC

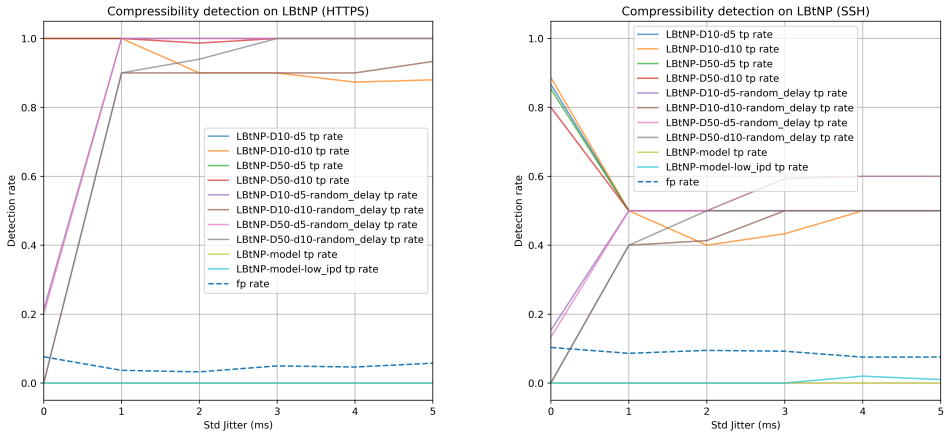


Figure 24: Compressibility detection on LBtNP

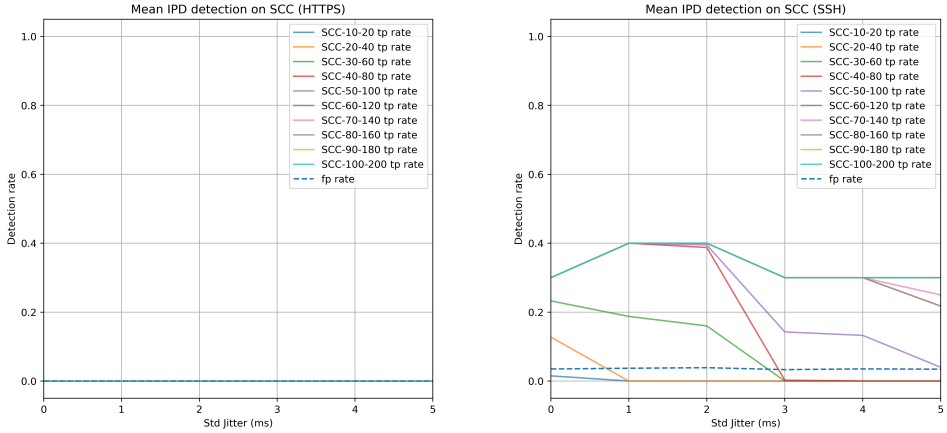


Figure 25: Mean IPD detection on SCC

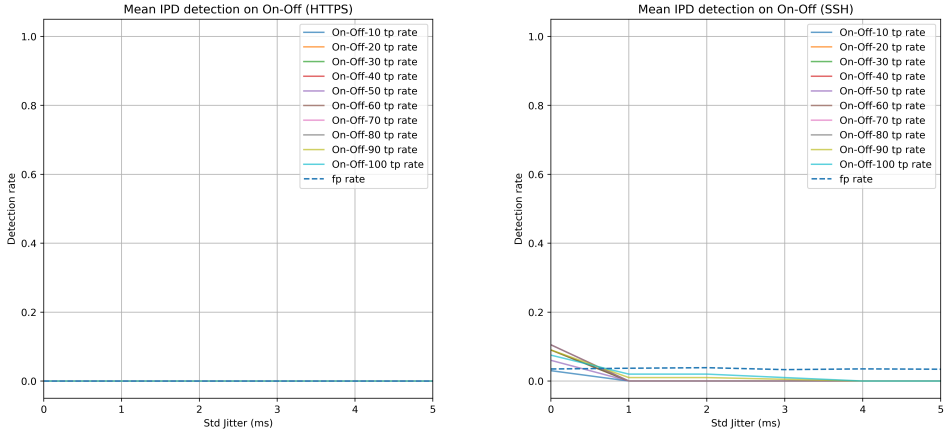


Figure 26: Mean IPD detection on On-Off

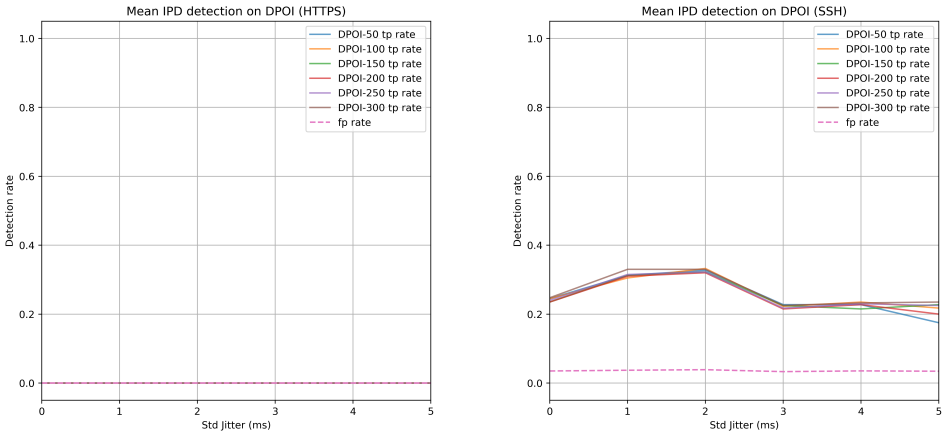


Figure 27: Mean IPD detection on DPOI

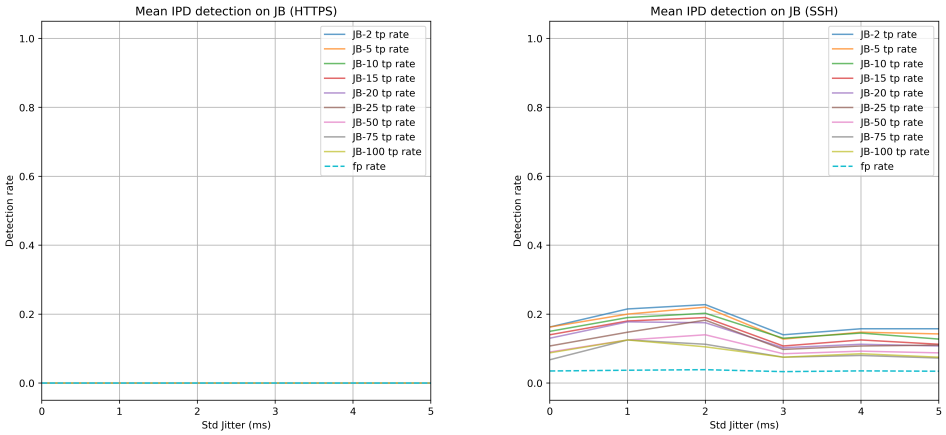


Figure 28: Mean IPD detection on JB

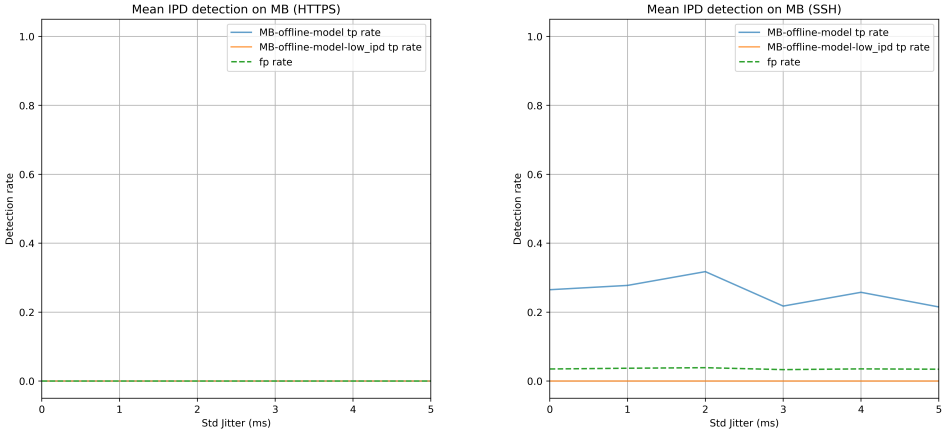


Figure 29: Mean IPD detection on MB-CTC

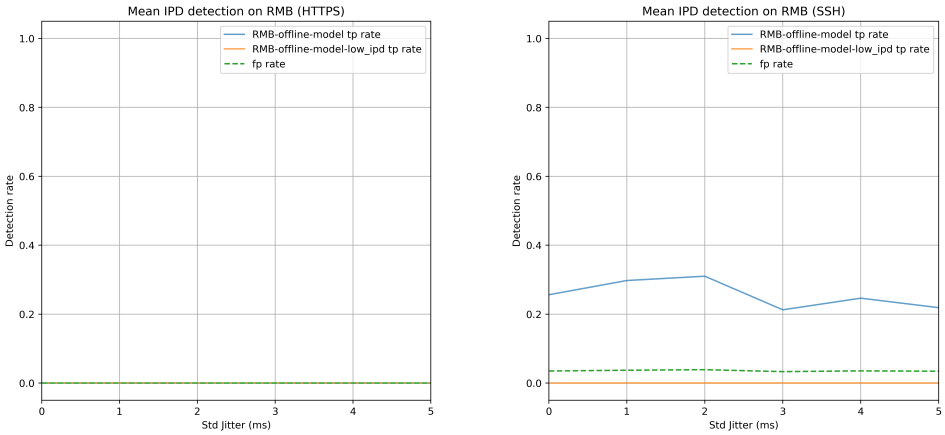


Figure 30: Mean IPD detection on RMB-CTC

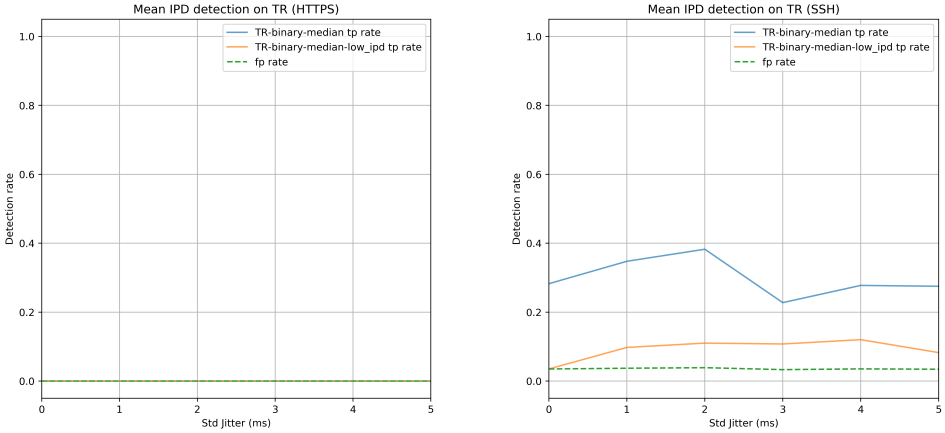


Figure 31: Mean IPD detection on TR-CTC

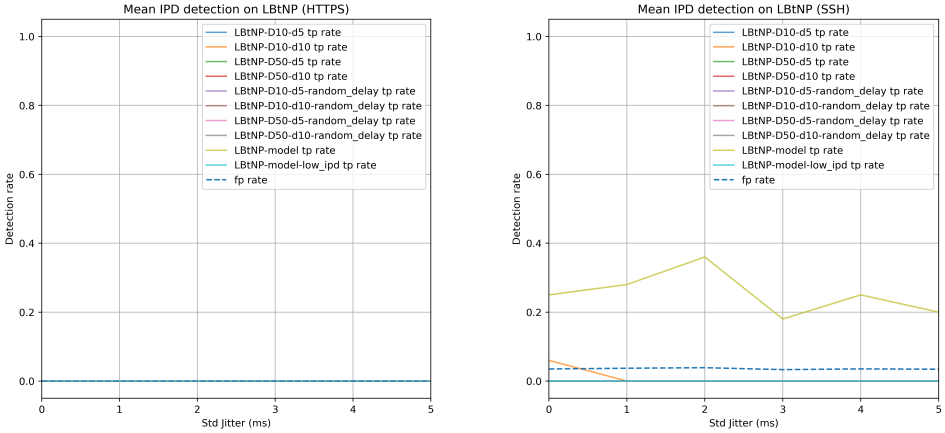


Figure 32: Mean IPD detection on LBtNP

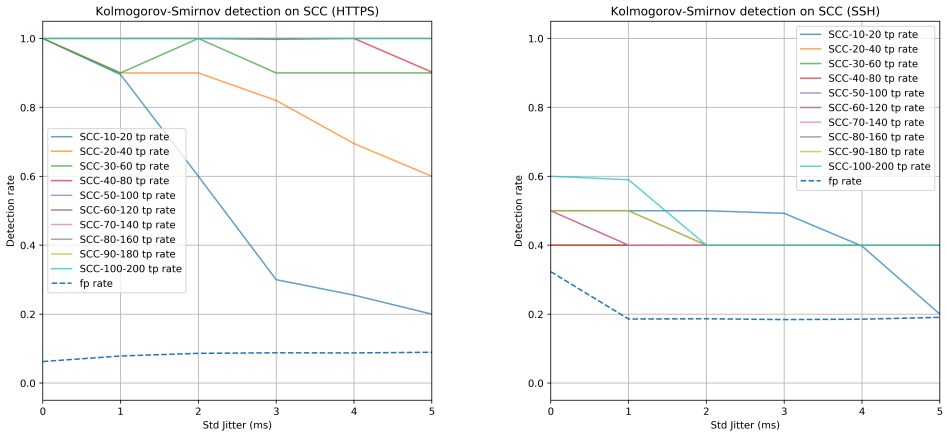


Figure 33: Kolmogorov-Smirnov detection on SCC

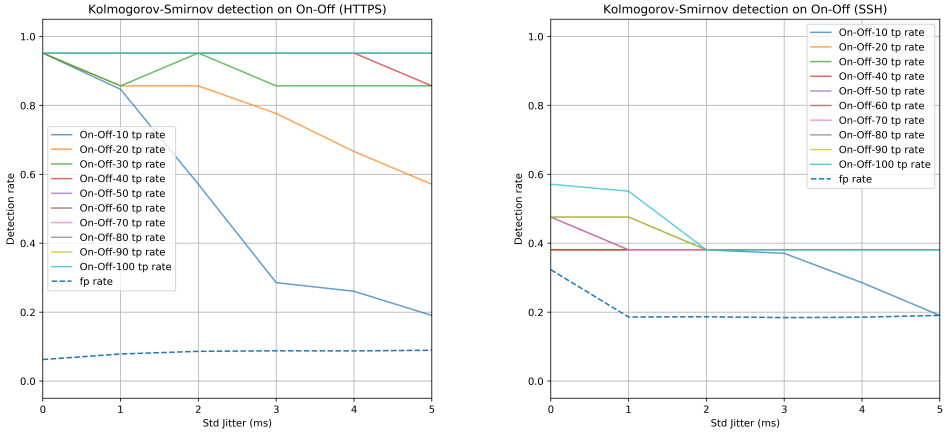


Figure 34: Kolmogorov-Smirnov detection on On-Off

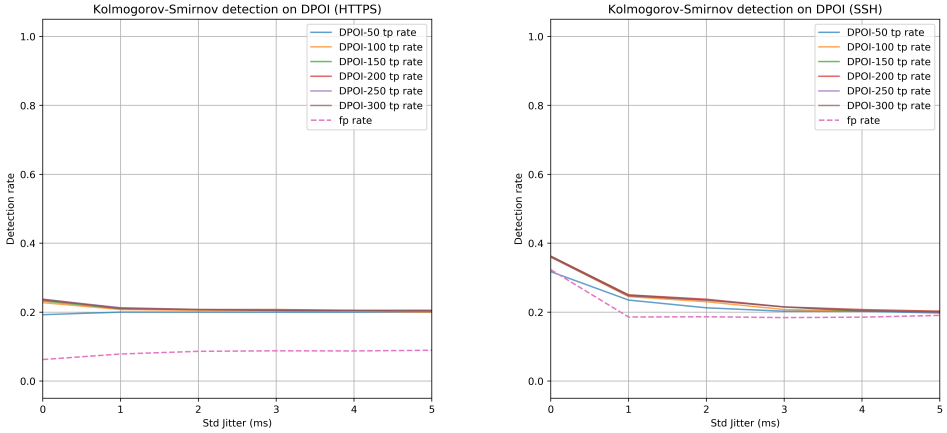


Figure 35: Kolmogorov-Smirnov detection on DPOI

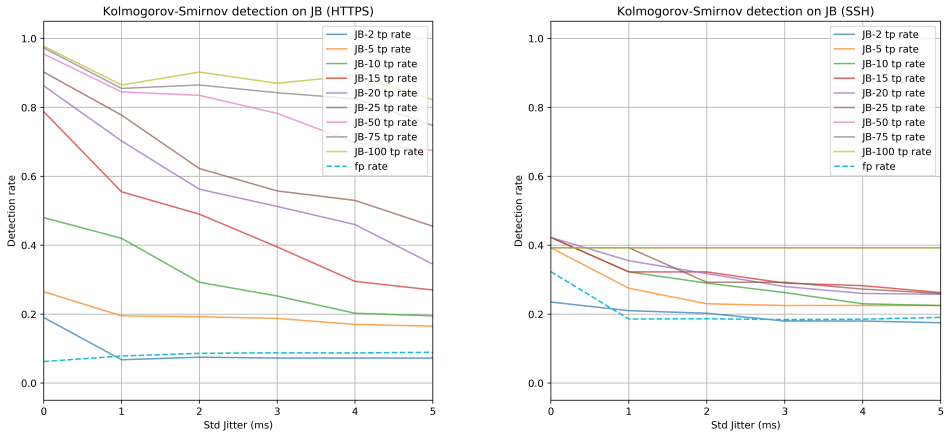


Figure 36: Kolmogorov-Smirnov detection on JB

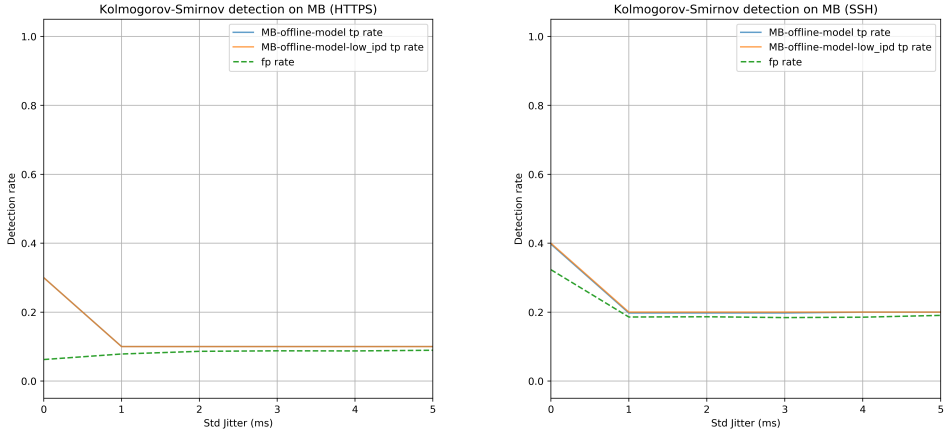


Figure 37: Kolmogorov-Smirnov detection on MB-CTC

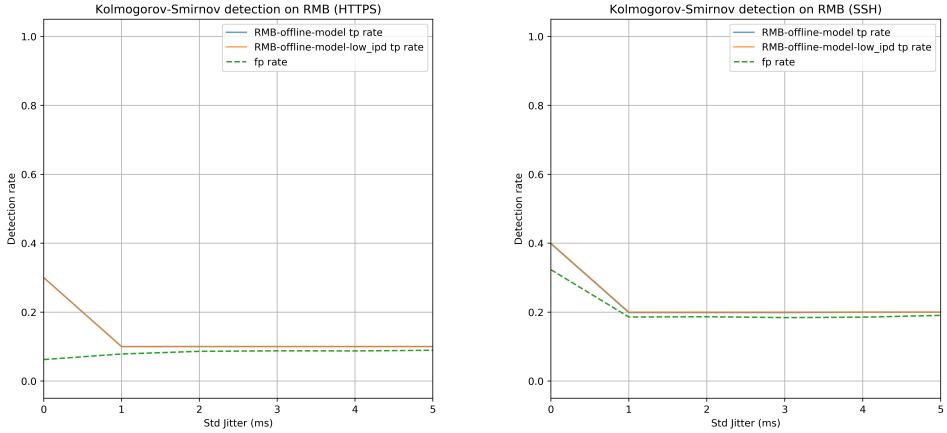


Figure 38: Kolmogorov-Smirnov detection on RMB-CTC

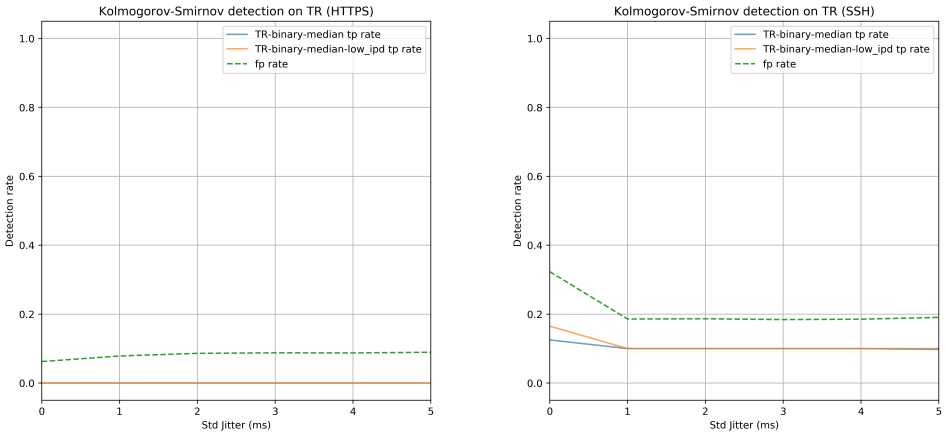


Figure 39: Kolmogorov-Smirnov detection on TR-CTC

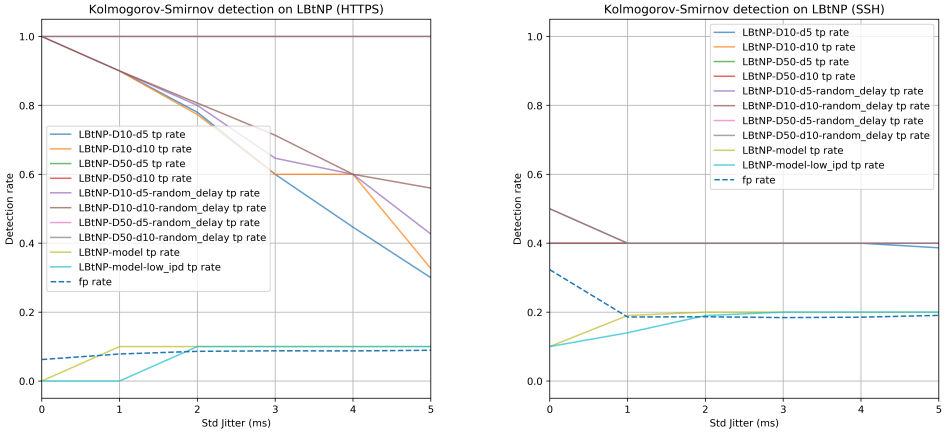


Figure 40: Kolmogorov-Smirnov detection on LBtNP

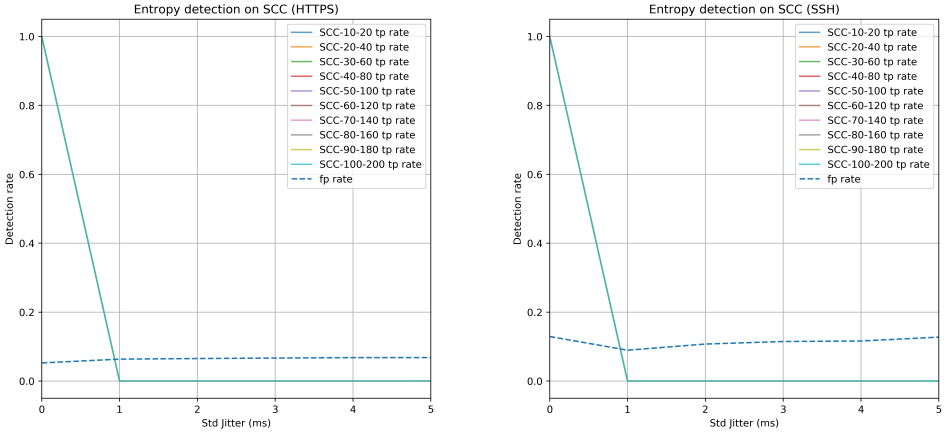


Figure 41: Entropy detection on SCC

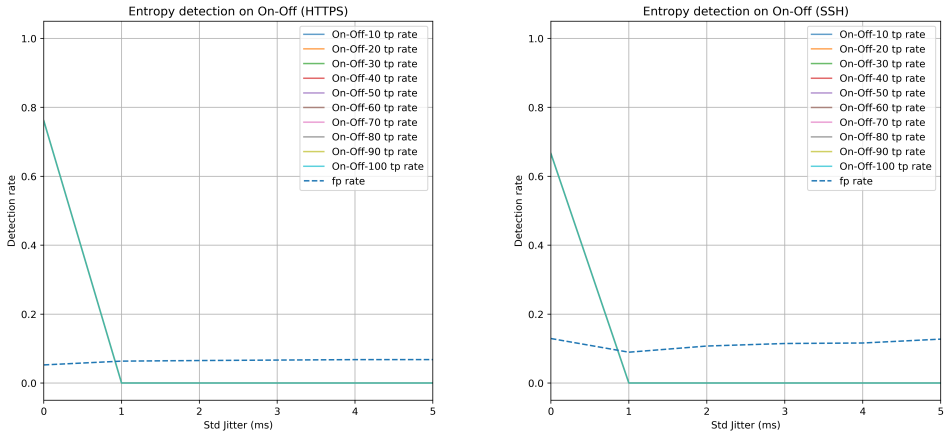


Figure 42: Entropy detection on On-Off

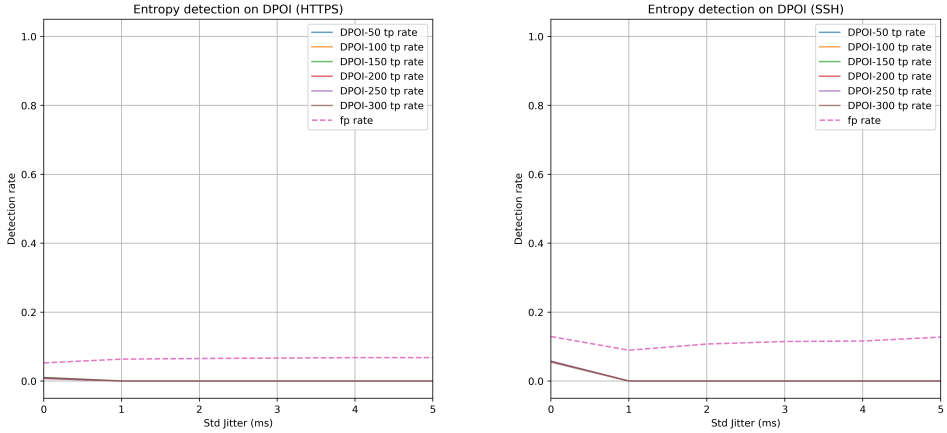


Figure 43: Entropy detection on DPOI

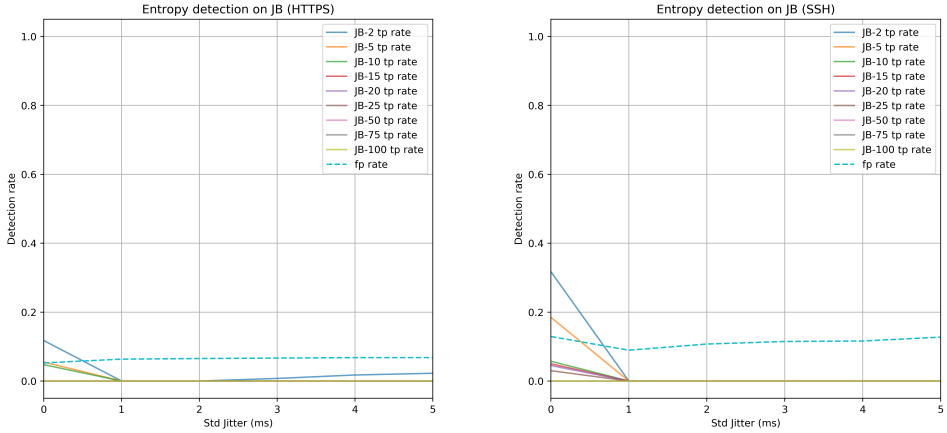


Figure 44: Entropy detection on JB

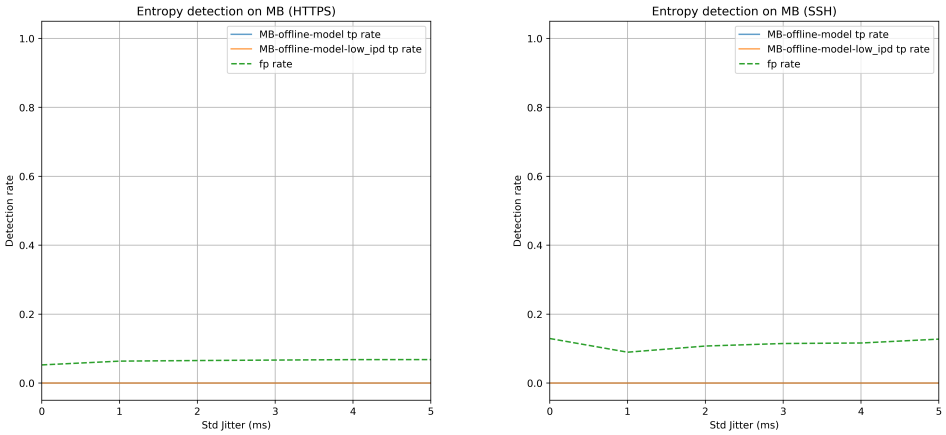


Figure 45: Entropy detection on MB-CTC

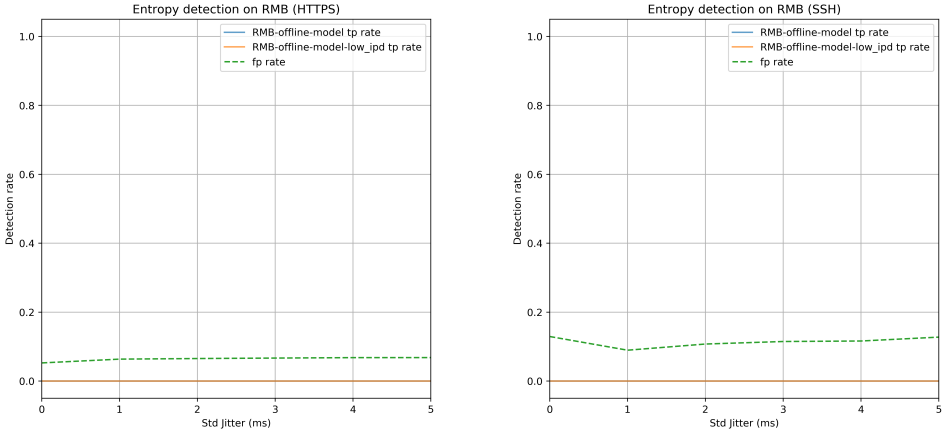


Figure 46: Entropy detection on RMB-CTC

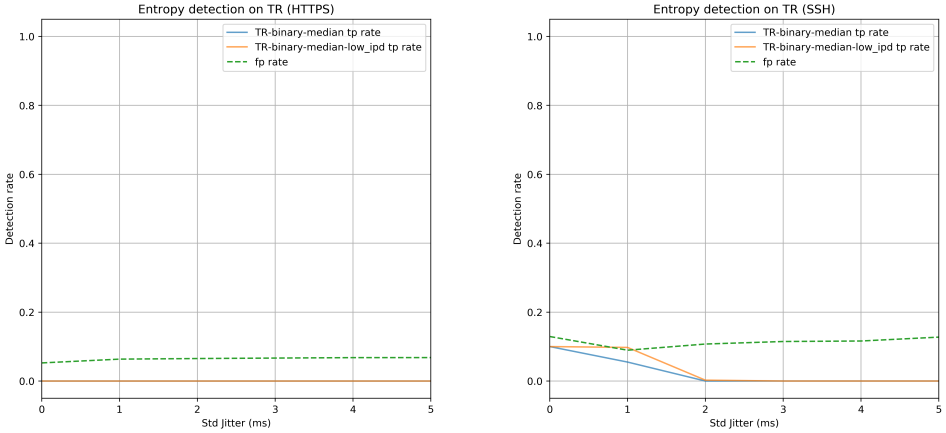


Figure 47: Entropy detection on TR-CTC

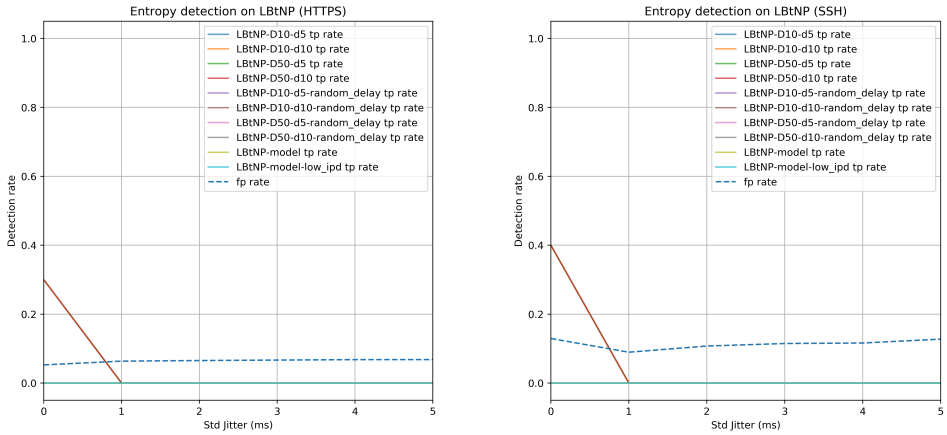


Figure 48: Entropy detection on LBtNP

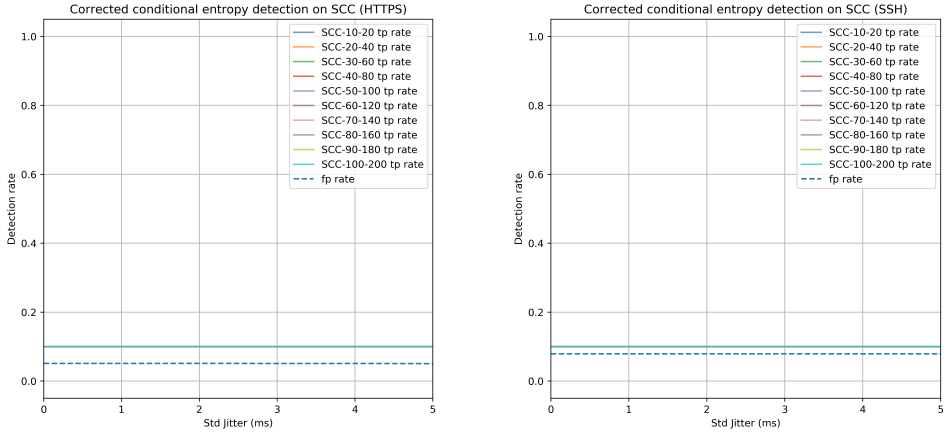


Figure 49: Corrected conditional entropy detection on SCC

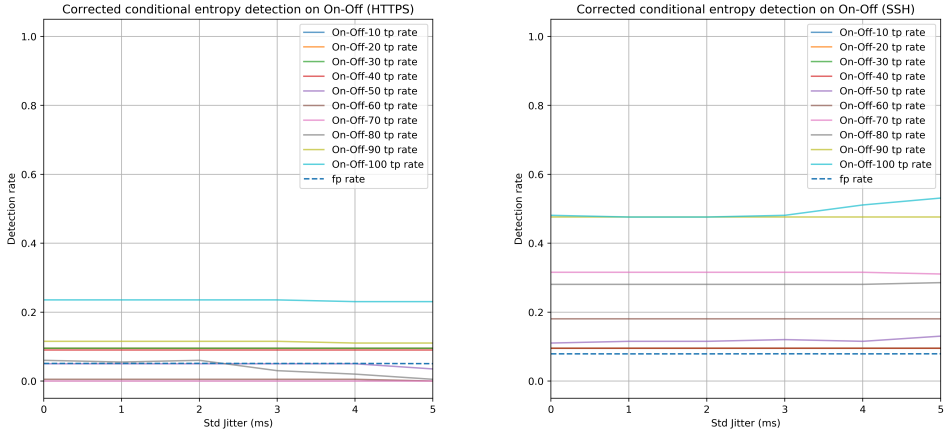


Figure 50: Corrected conditional entropy detection on On-Off

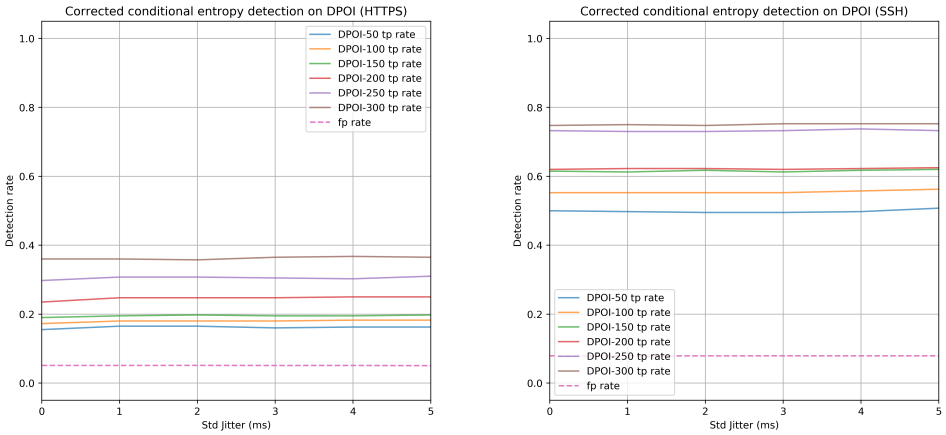


Figure 51: Corrected conditional entropy detection on DPOI

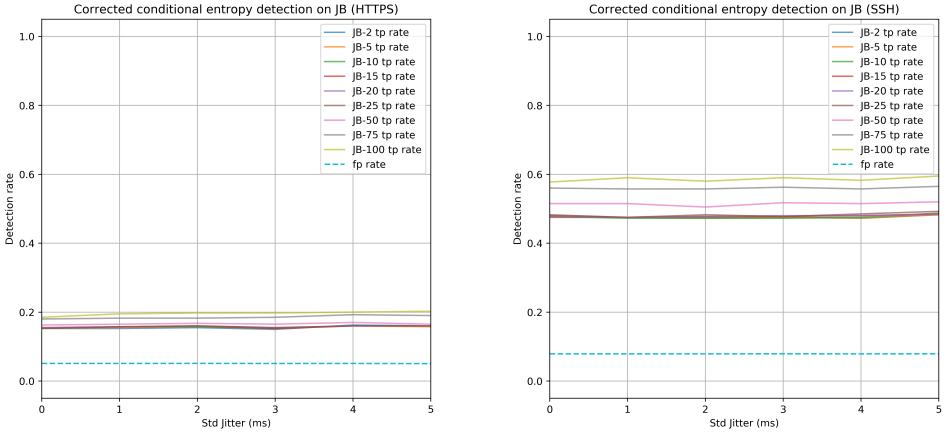


Figure 52: Corrected conditional entropy detection on JB

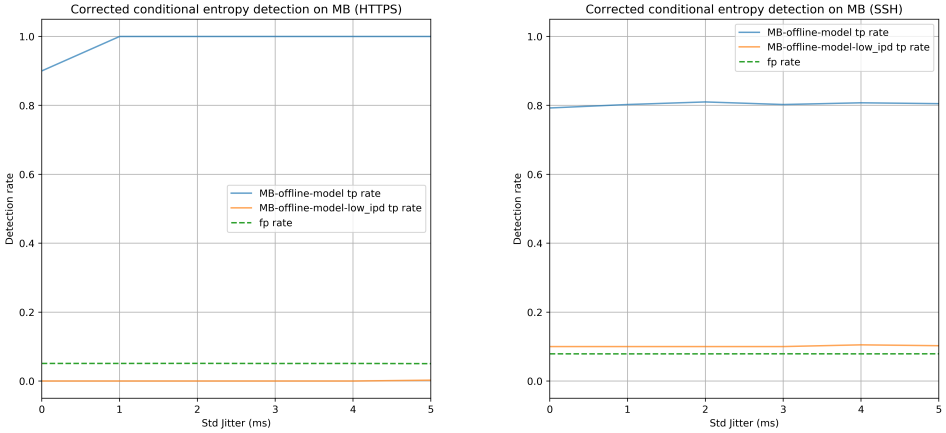


Figure 53: Corrected conditional entropy detection on MB-CTC

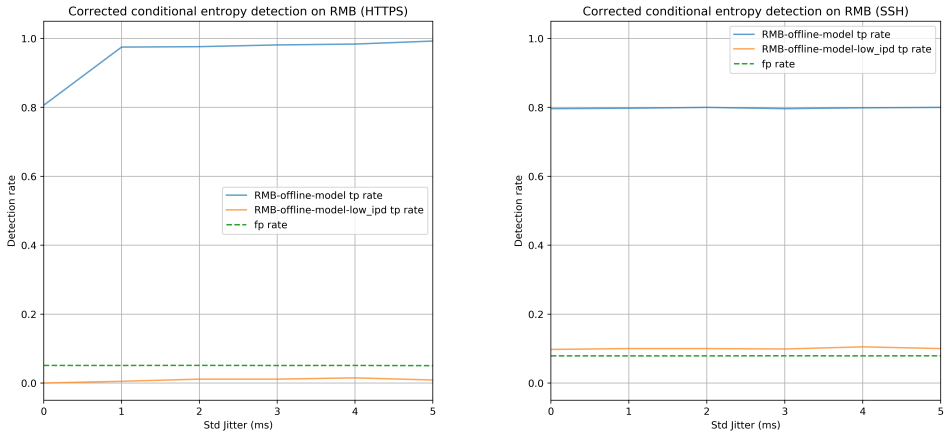


Figure 54: Corrected conditional entropy detection on RMB-CTC

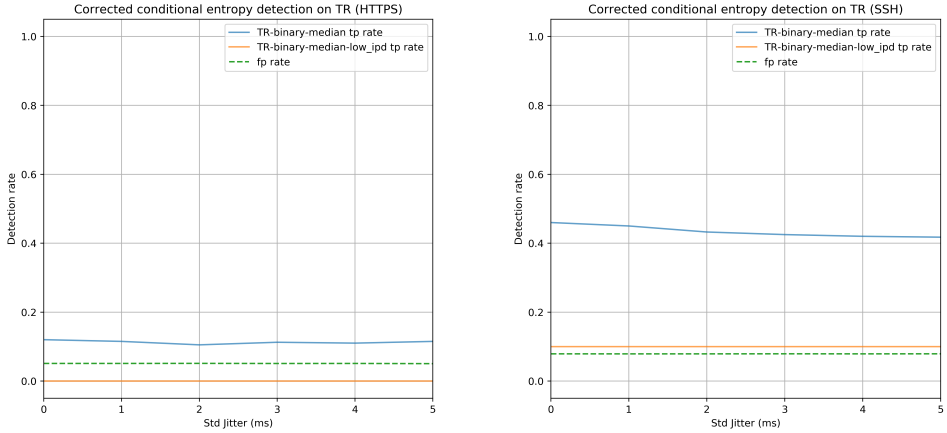


Figure 55: Corrected conditional entropy detection on TR-CTC

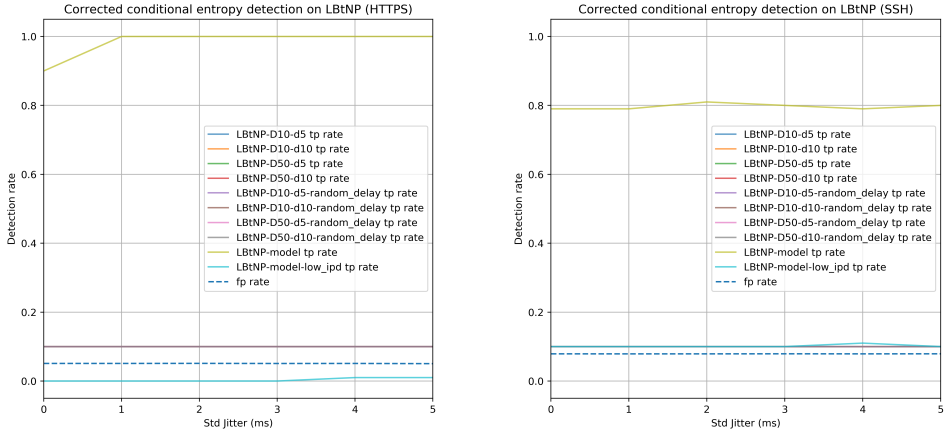


Figure 56: Corrected conditional entropy detection on LBtNP

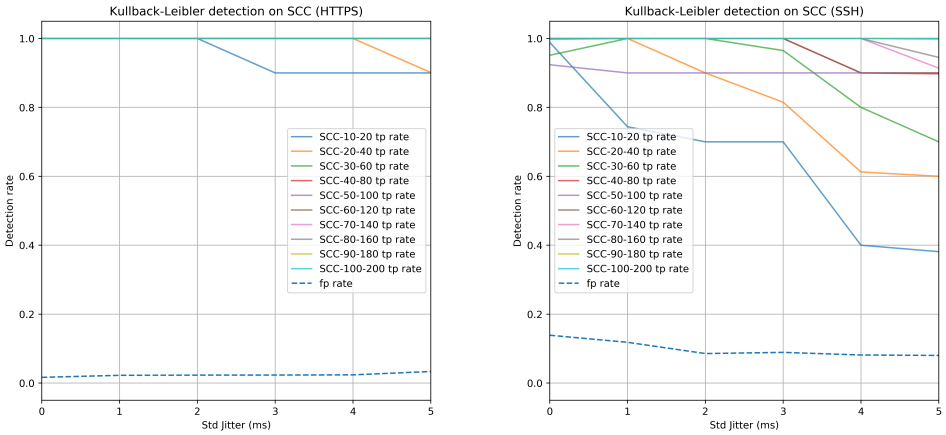


Figure 57: Kullback-Leibler detection on SCC

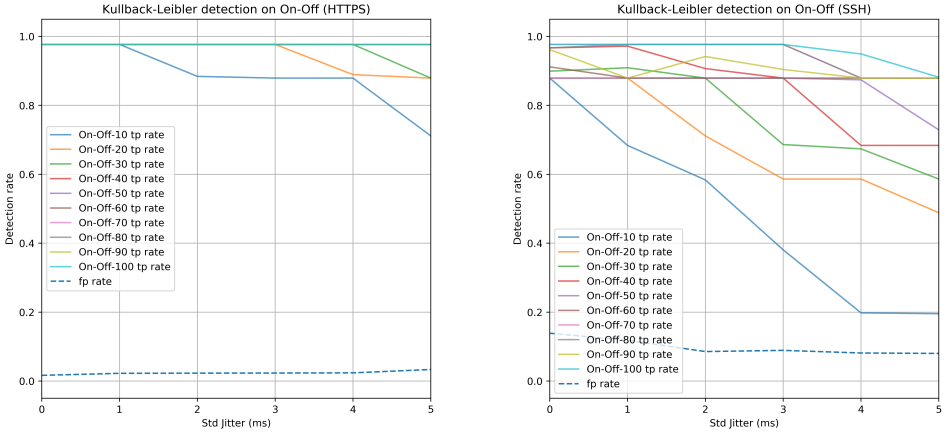


Figure 58: Kullback-Leibler detection on On-Off

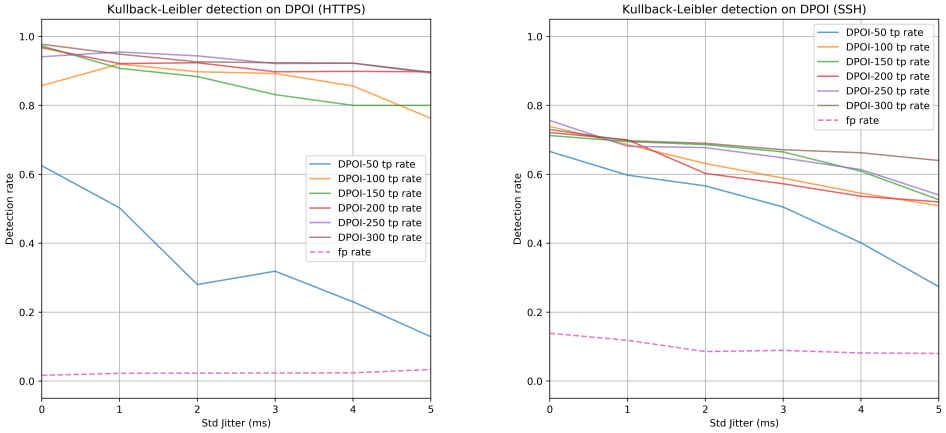


Figure 59: Kullback-Leibler detection on DPOI

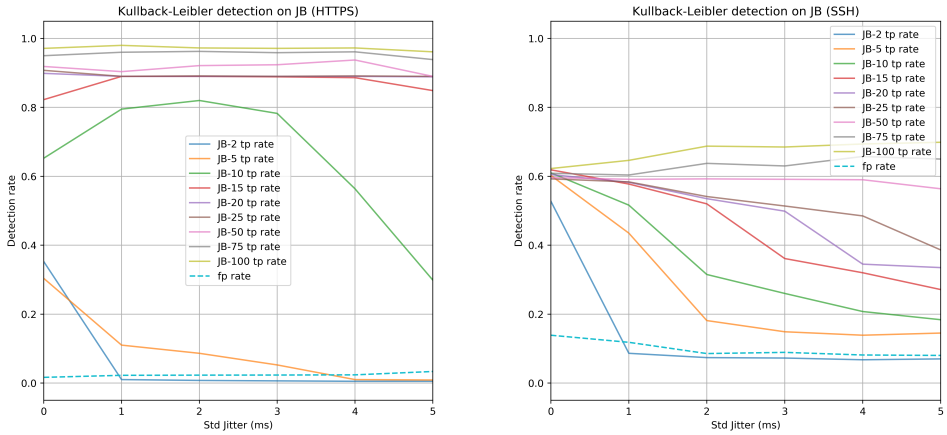


Figure 60: Kullback-Leibler detection on JB

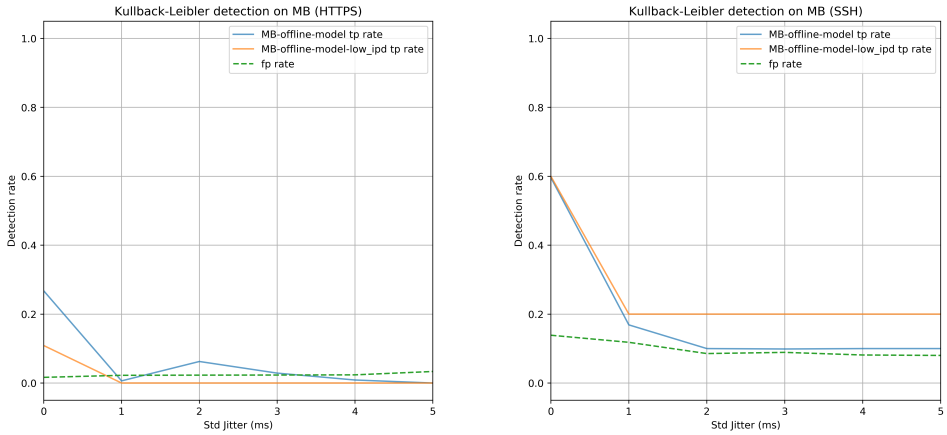


Figure 61: Kullback-Leibler detection on MB-CTC

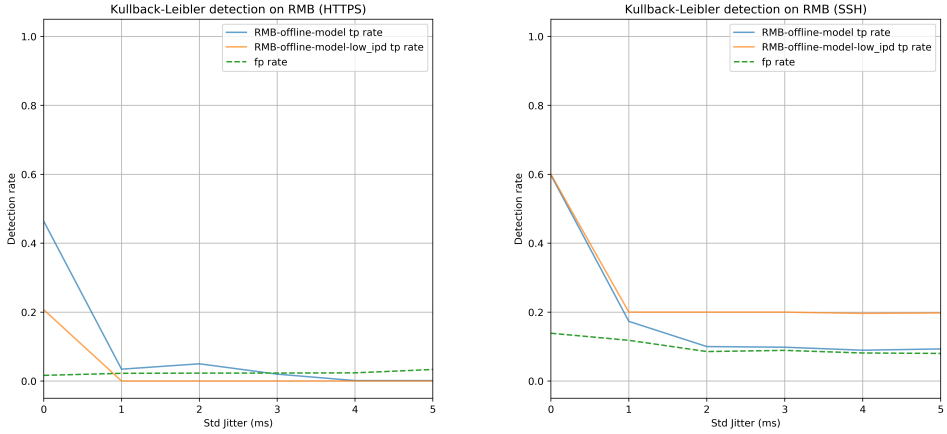


Figure 62: Kullback-Leibler detection on RMB-CTC

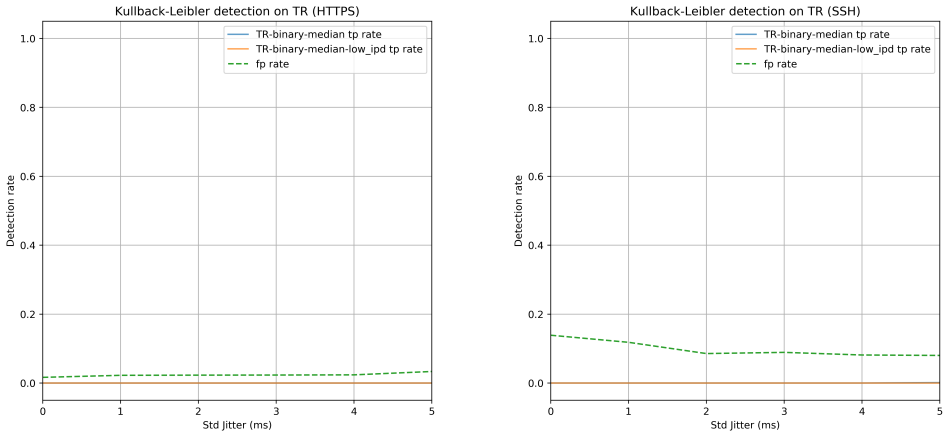


Figure 63: Kullback-Leibler detection on TR-CTC

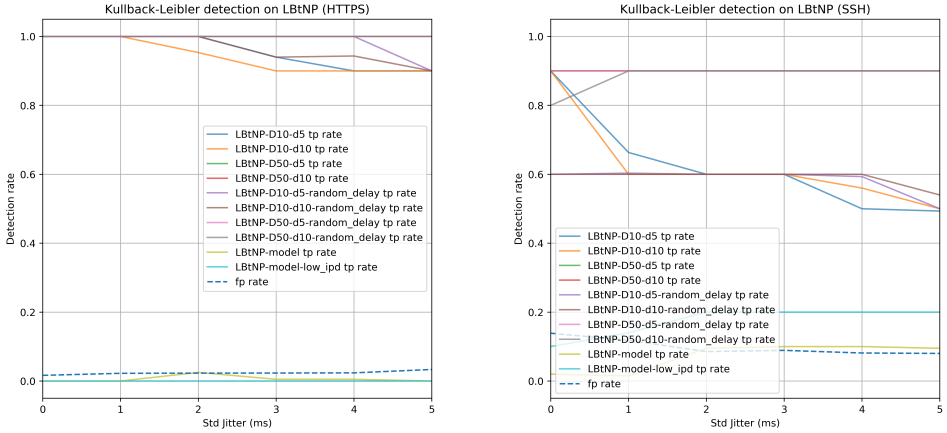


Figure 64: Kullback-Leibler detection on LBtNP

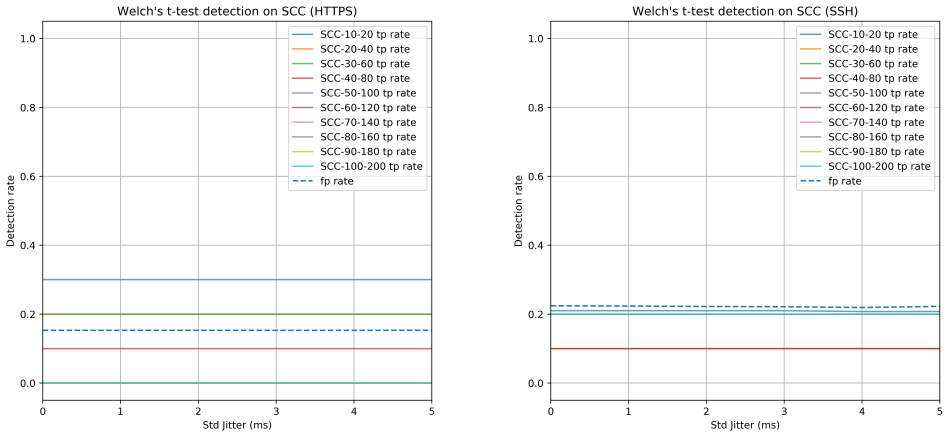


Figure 65: Welch's t-test detection on SCC

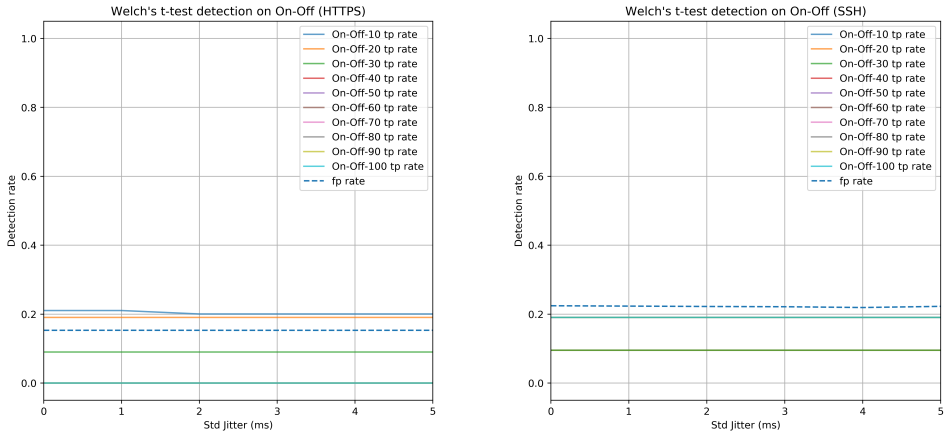


Figure 66: Welch's t-test detection on On-Off

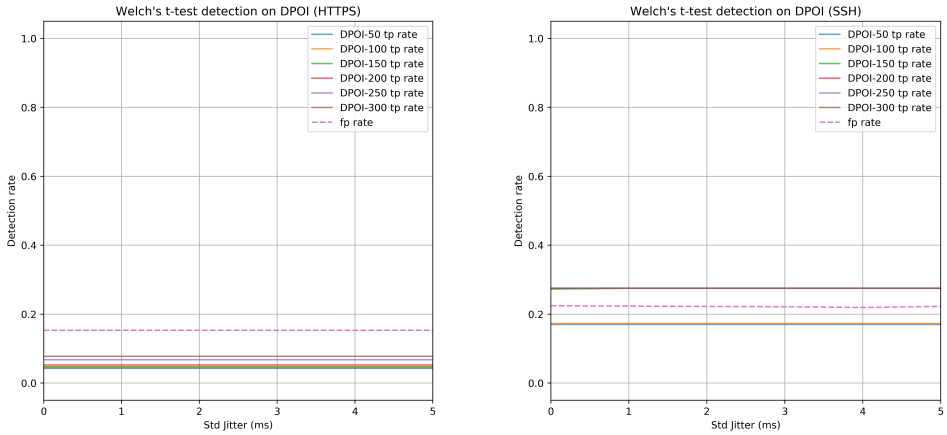


Figure 67: Welch's t-test detection on DPOI

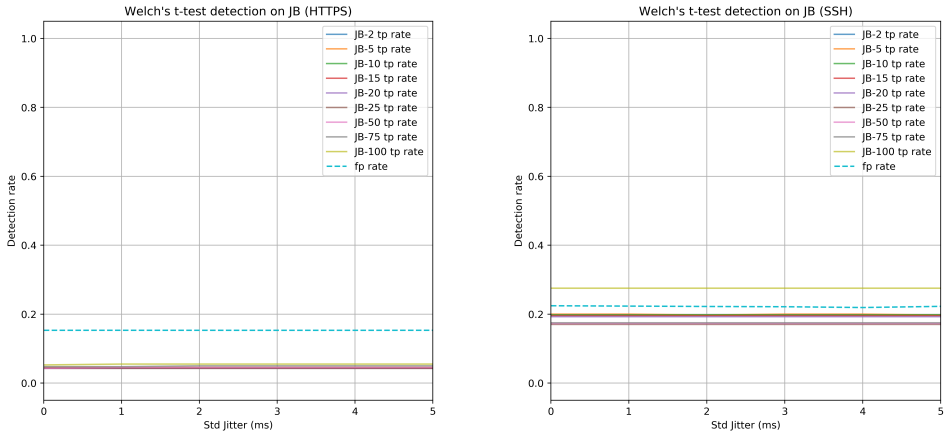


Figure 68: Welch's t-test detection on JB

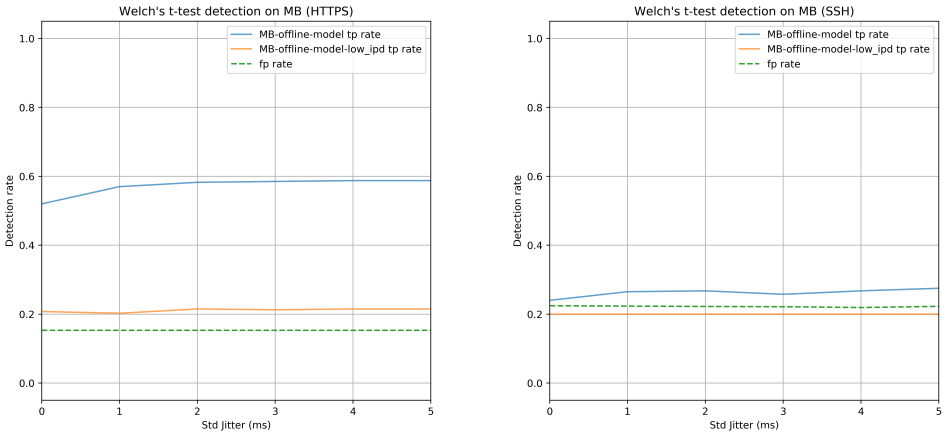


Figure 69: Welch's t-test detection on MB-CTC

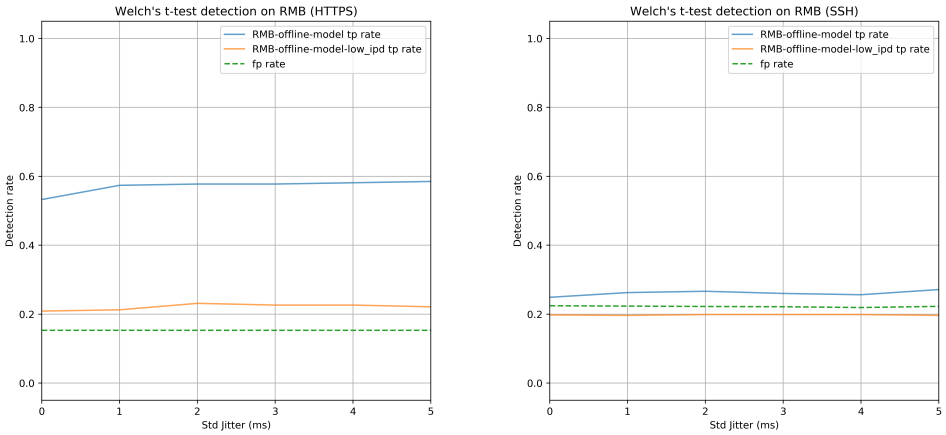


Figure 70: Welch's t-test detection on RMB-CTC

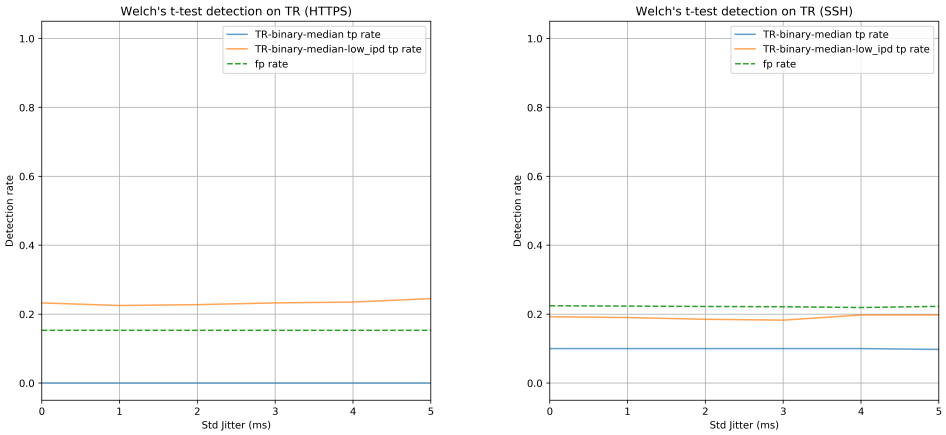


Figure 71: Welch's t-test detection on TR-CTC

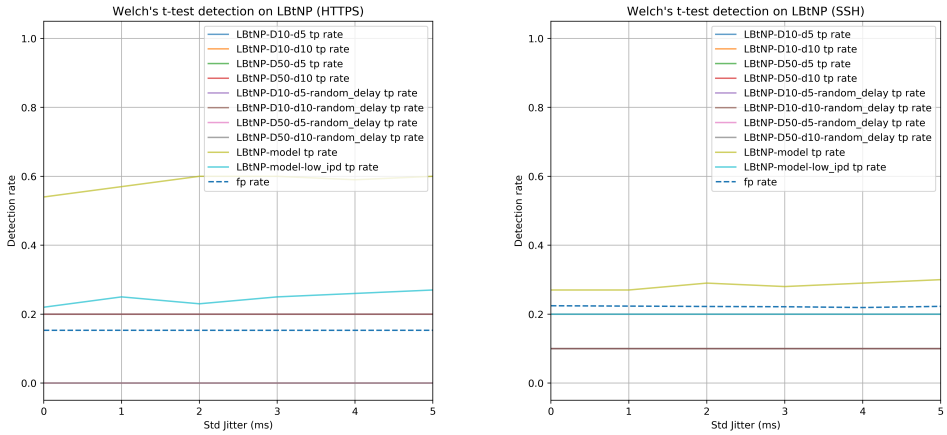


Figure 72: Welch's t-test detection on LBtNP

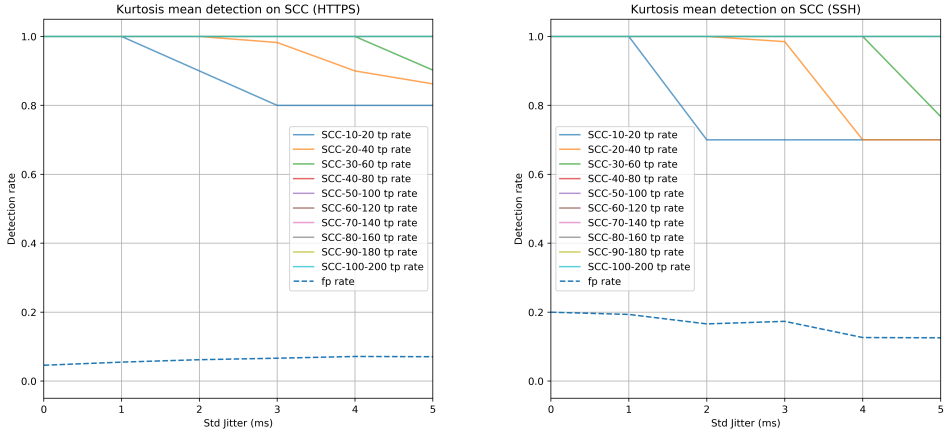


Figure 73: Kurtosis mean detection on SCC

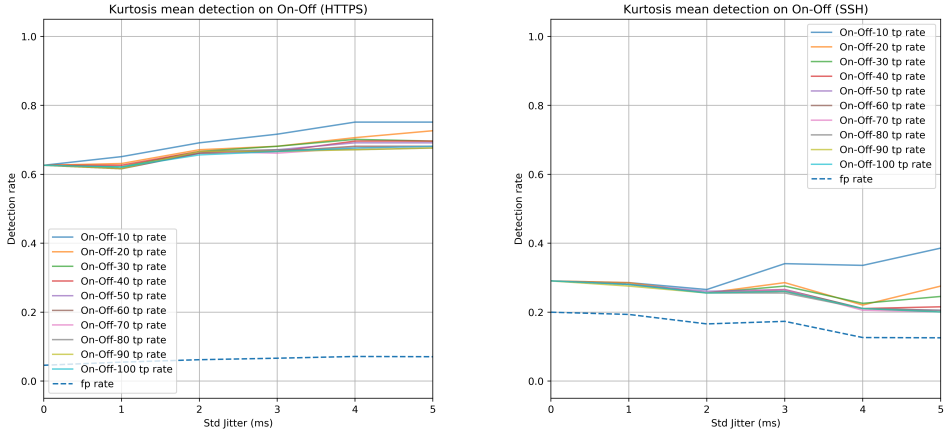


Figure 74: Kurtosis mean detection on On-Off

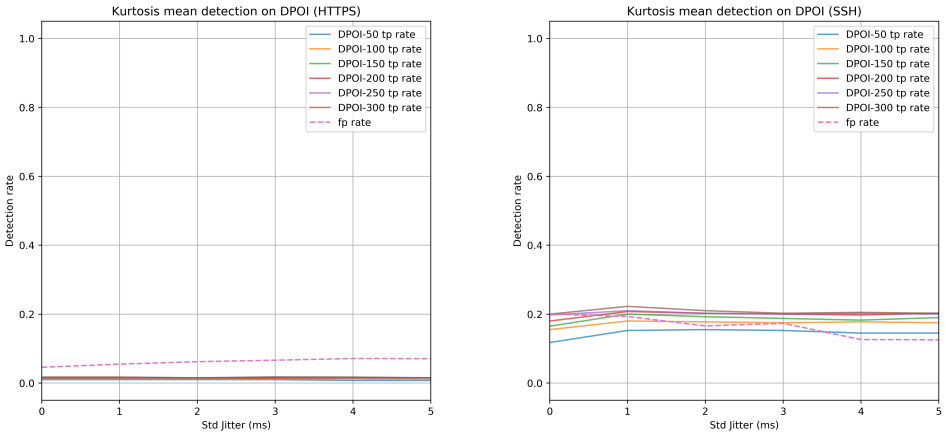


Figure 75: Kurtosis mean detection on DPOI

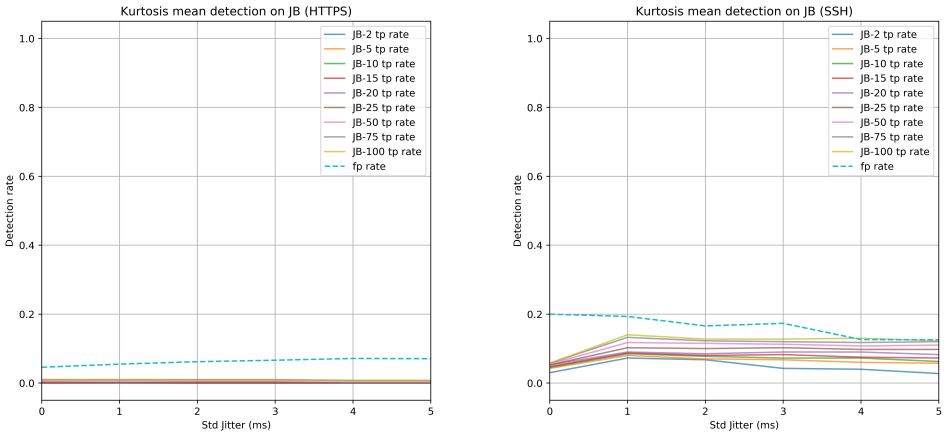


Figure 76: Kurtosis mean detection on JB

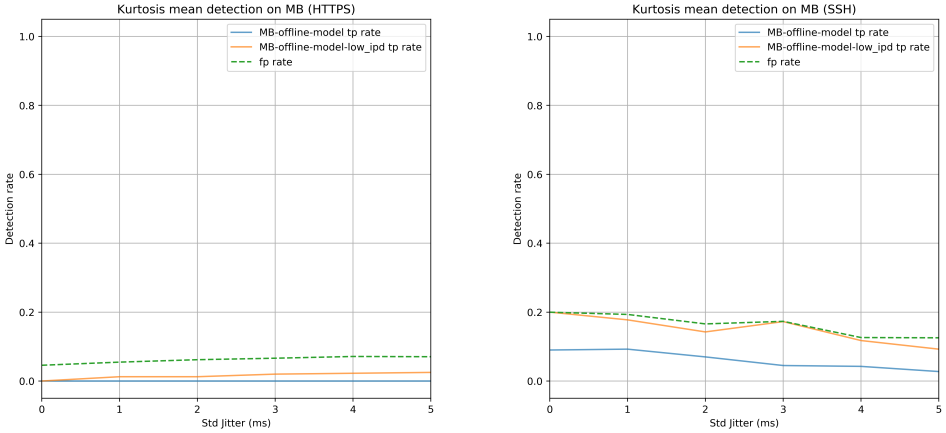


Figure 77: Kurtosis mean detection on MB-CTC

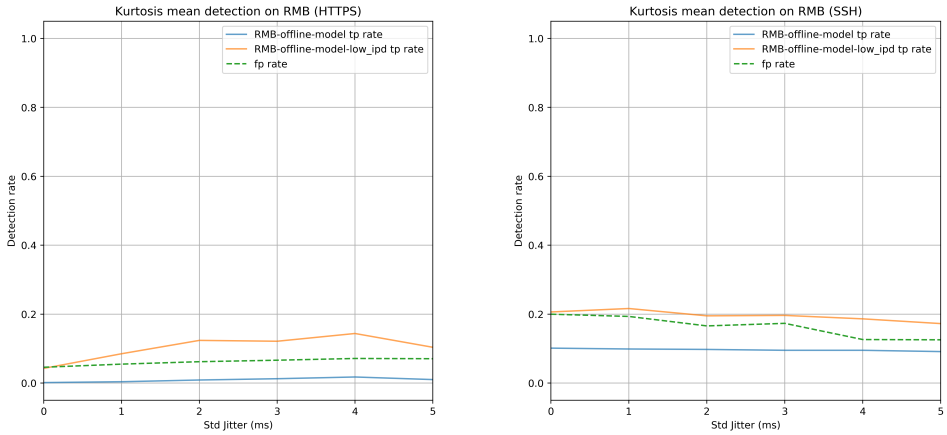


Figure 78: Kurtosis mean detection on RMB-CTC

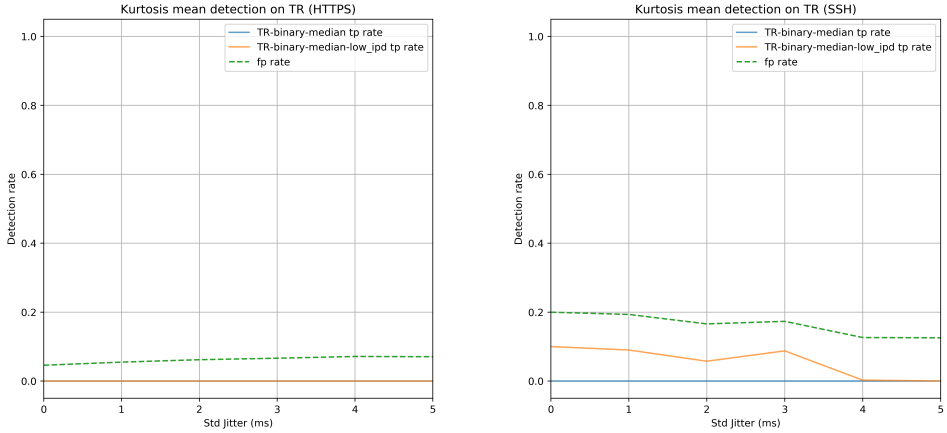


Figure 79: Kurtosis mean detection on TR-CTC

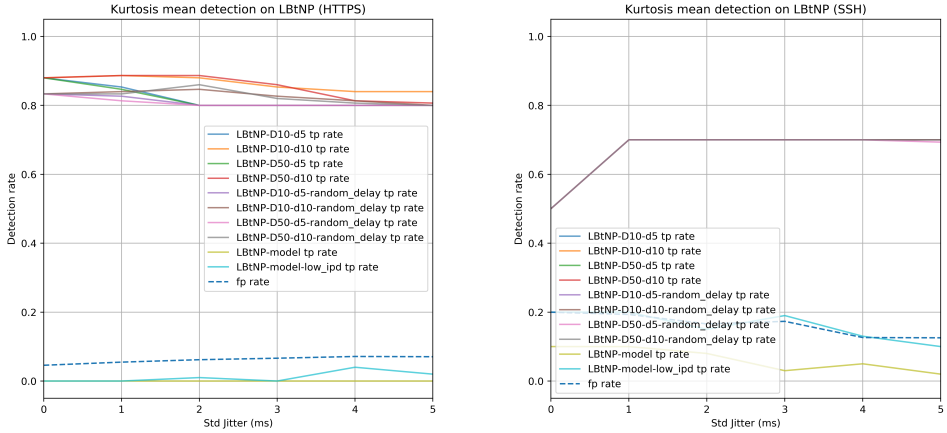


Figure 80: Kurtosis mean detection on LBtNP

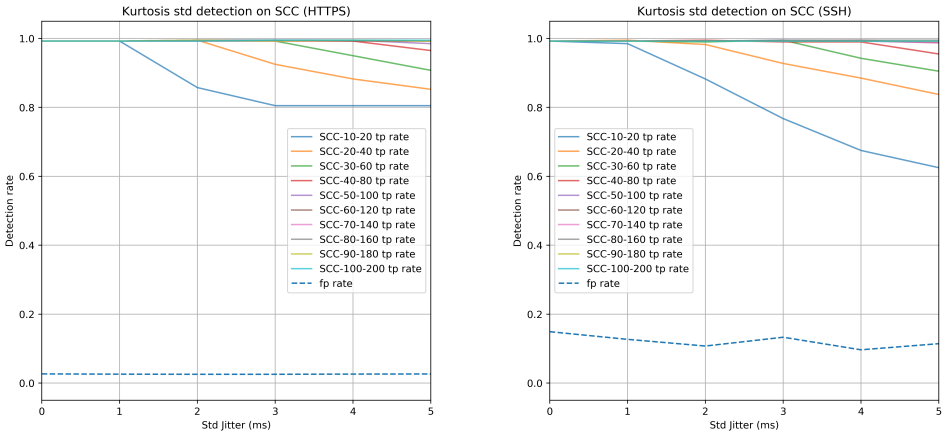


Figure 81: Kurtosis std detection on SCC

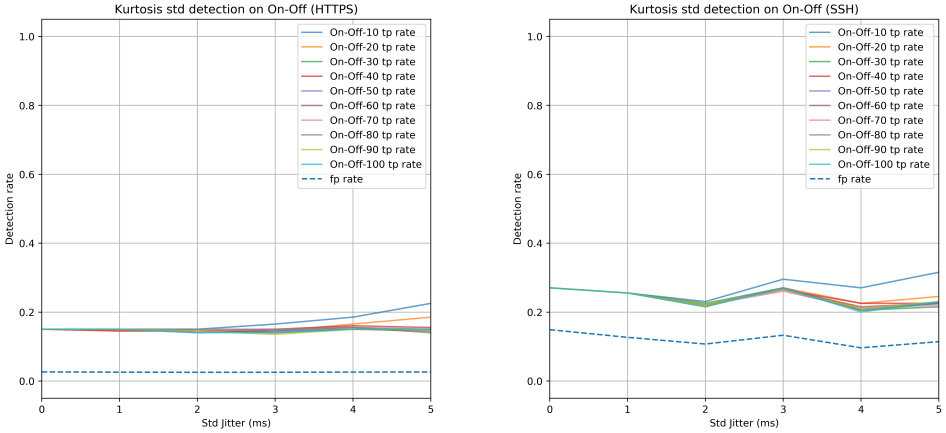


Figure 82: Kurtosis std detection on On-Off

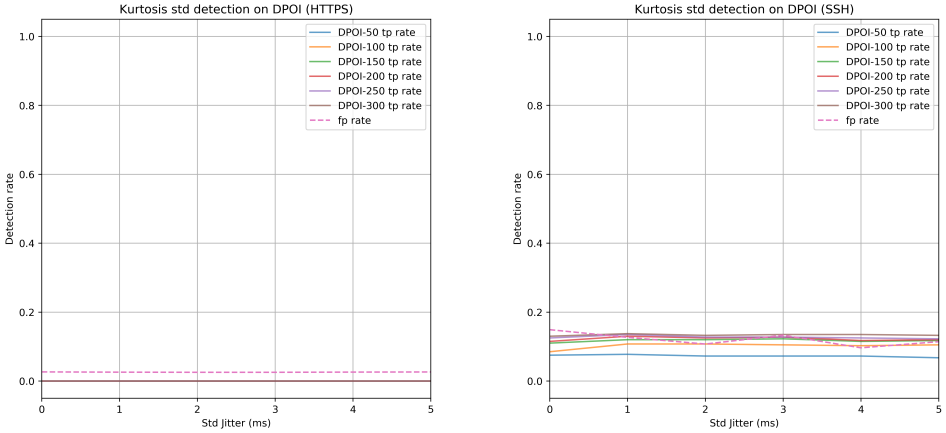


Figure 83: Kurtosis std detection on DPOI

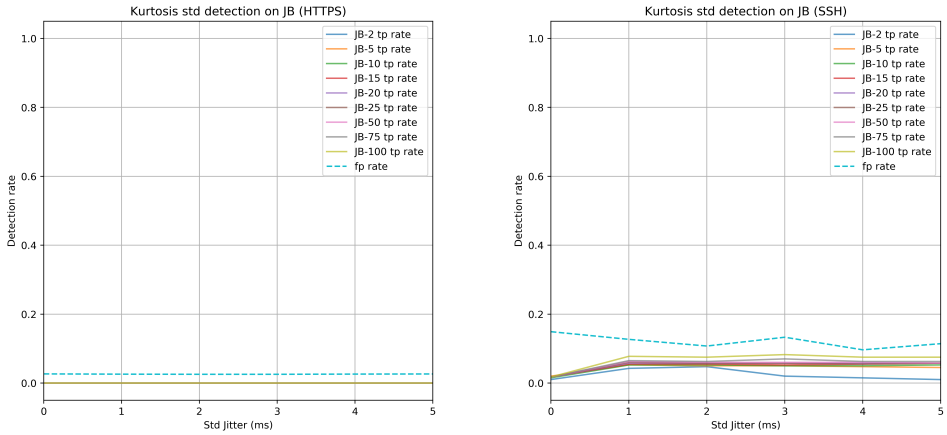


Figure 84: Kurtosis std detection on JB

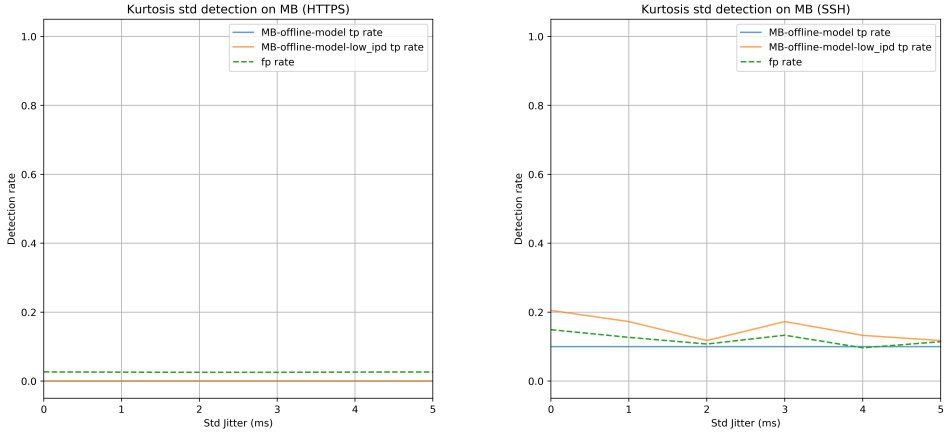


Figure 85: Kurtosis std detection on MB-CTC

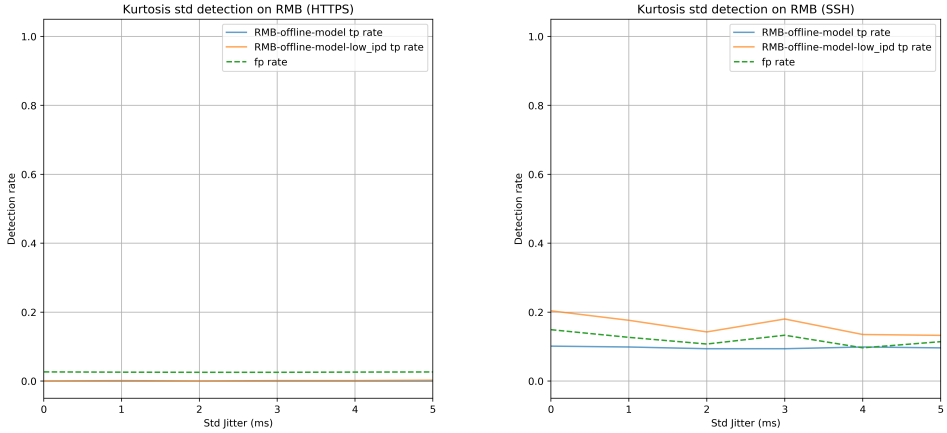


Figure 86: Kurtosis std detection on RMB-CTC

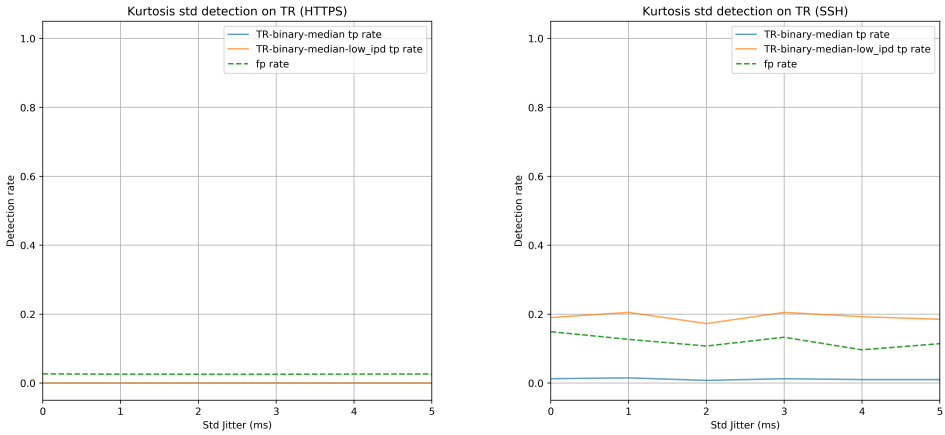


Figure 87: Kurtosis std detection on TR-CTC

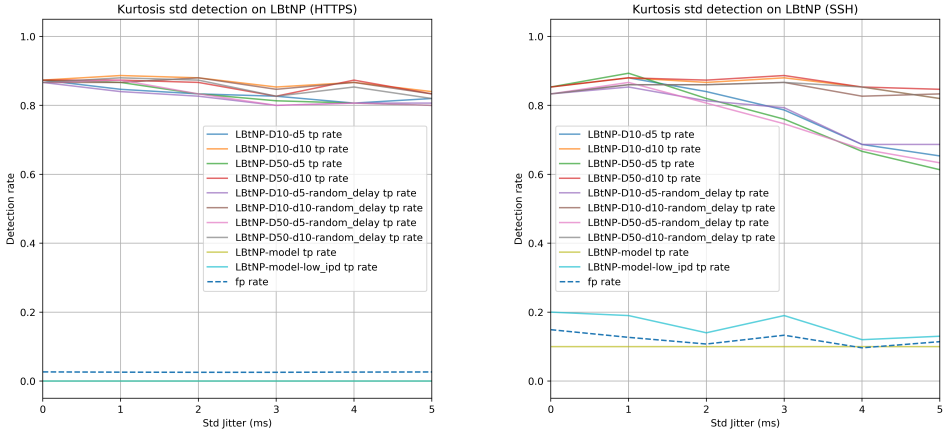


Figure 88: Kurtosis std detection on LBtNP

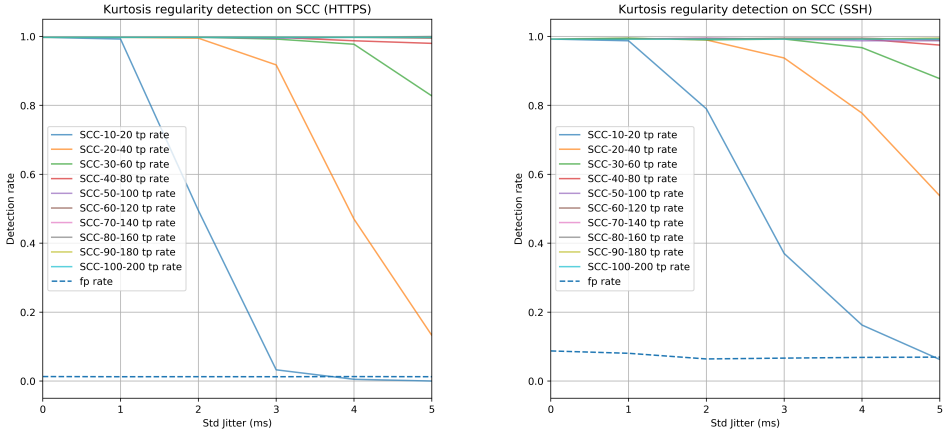


Figure 89: Kurtosis regularity detection on SCC

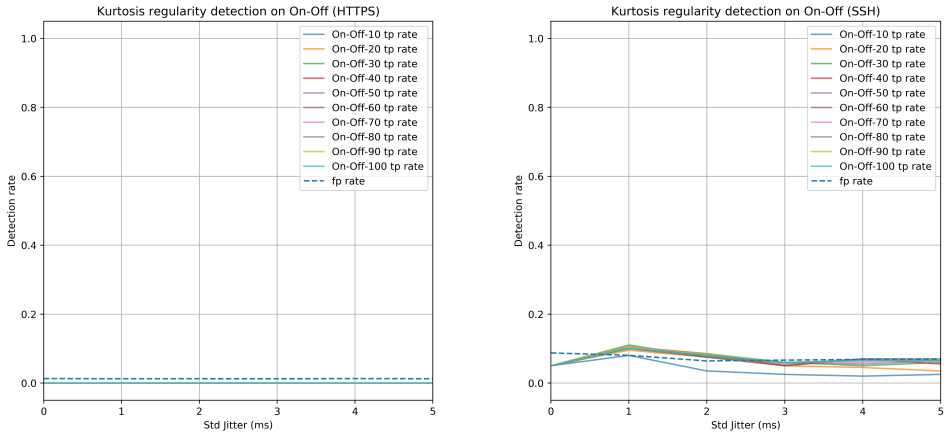


Figure 90: Kurtosis regularity detection on On-Off

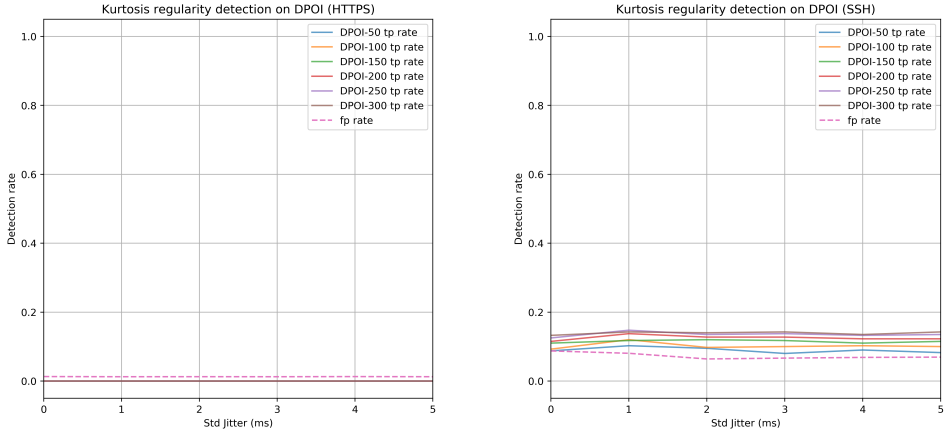


Figure 91: Kurtosis regularity detection on DPOI

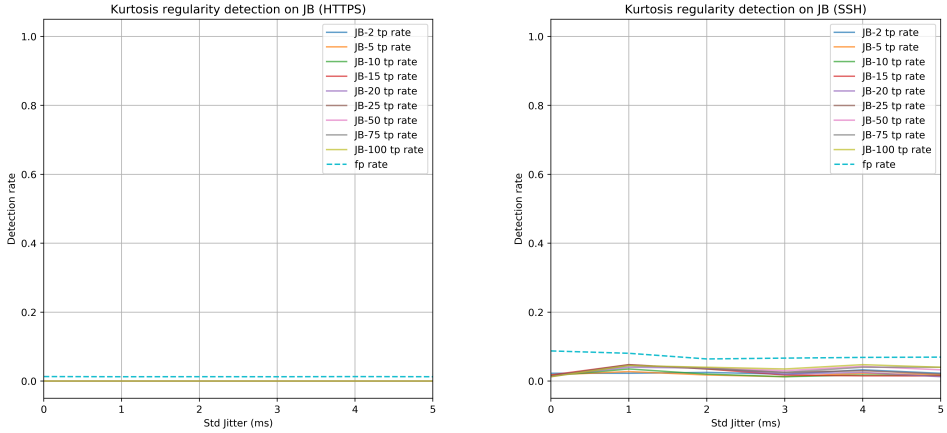


Figure 92: Kurtosis regularity detection on JB

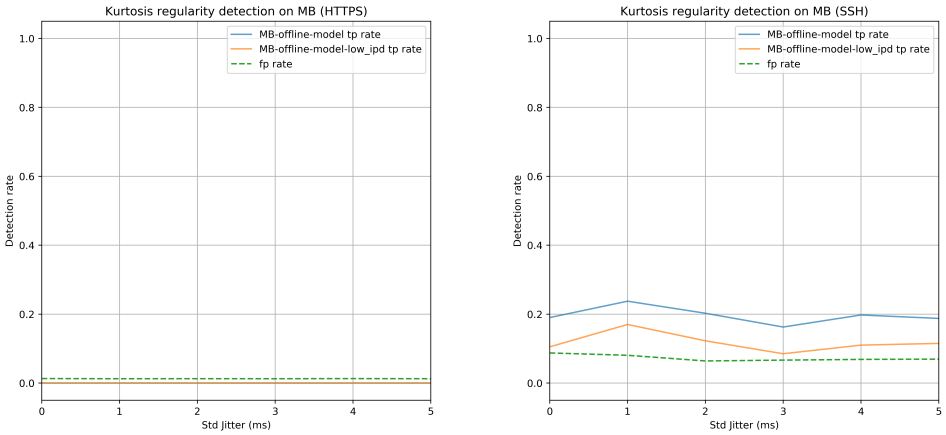


Figure 93: Kurtosis regularity detection on MB-CTC

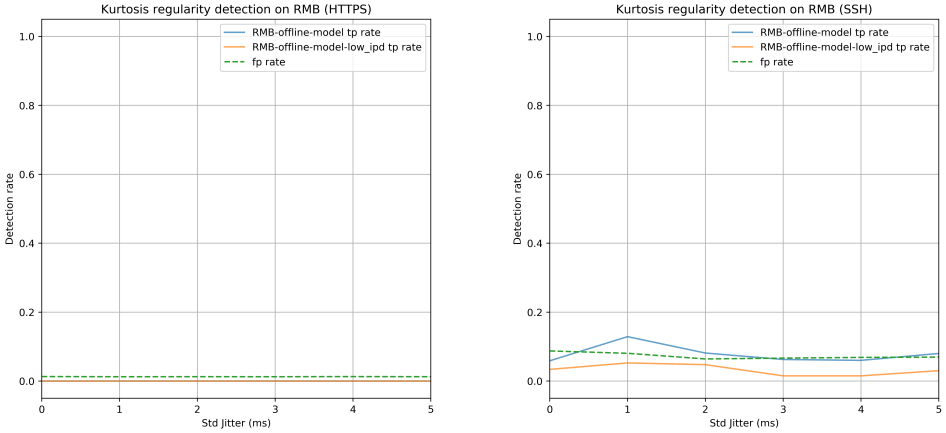


Figure 94: Kurtosis regularity detection on RMB-CTC

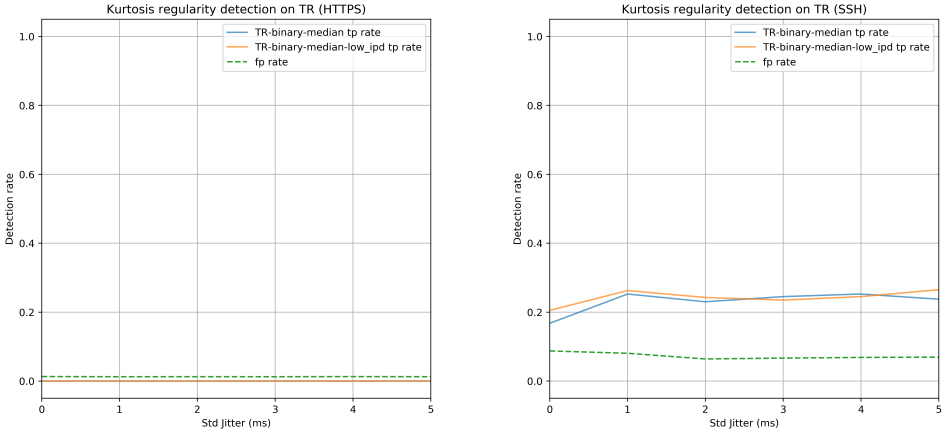


Figure 95: Kurtosis regularity detection on TR-CTC

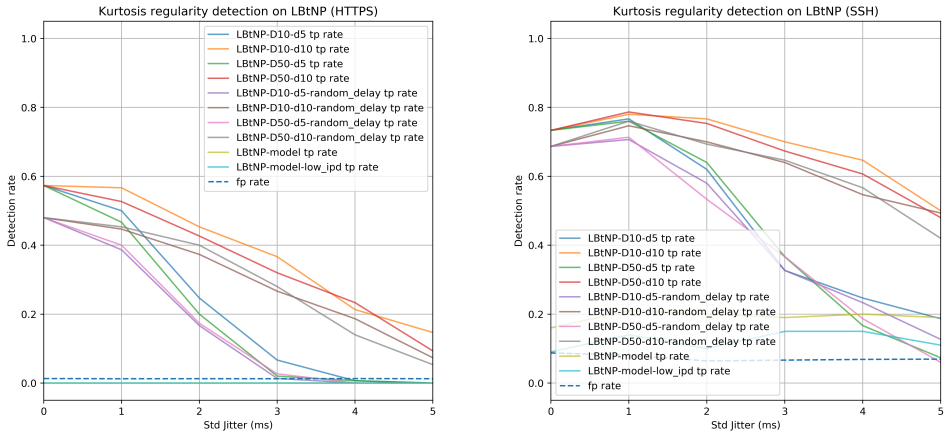


Figure 96: Kurtosis regularity detection on LBtNP

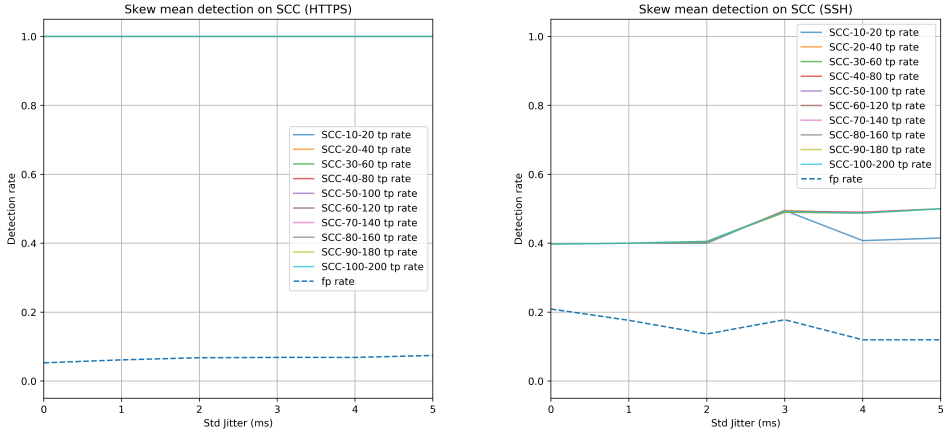


Figure 97: Skew mean detection on SCC

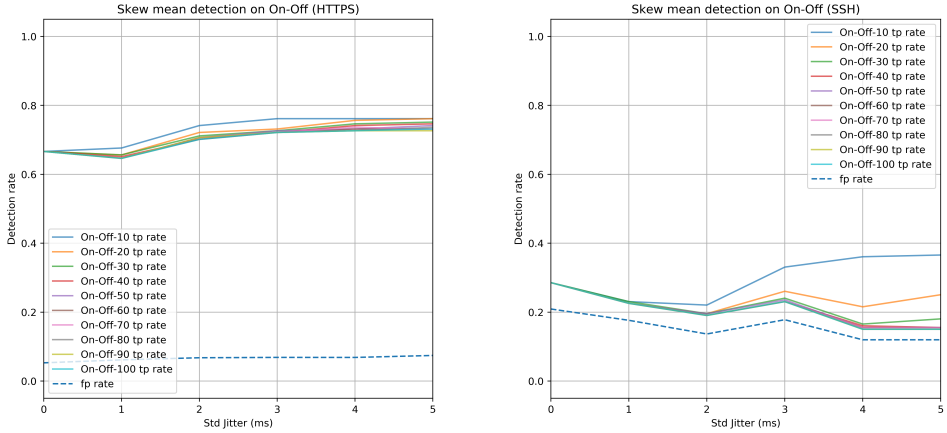


Figure 98: Skew mean detection on On-Off

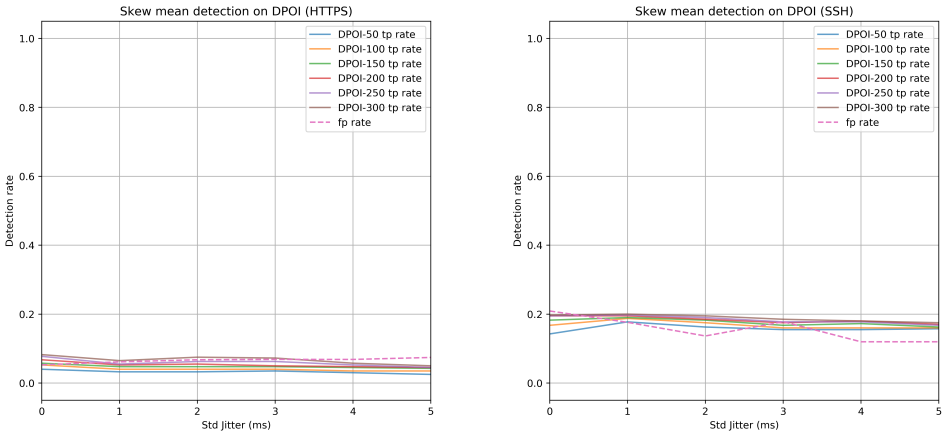


Figure 99: Skew mean detection on DPOI

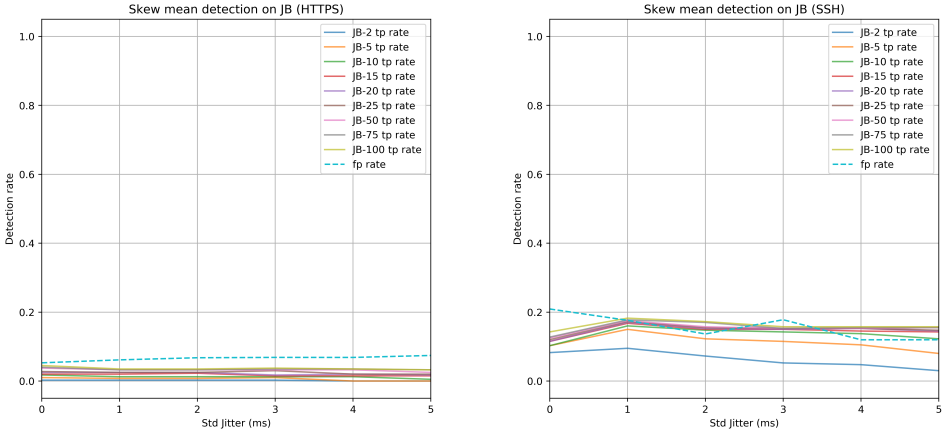


Figure 100: Skew mean detection on JB

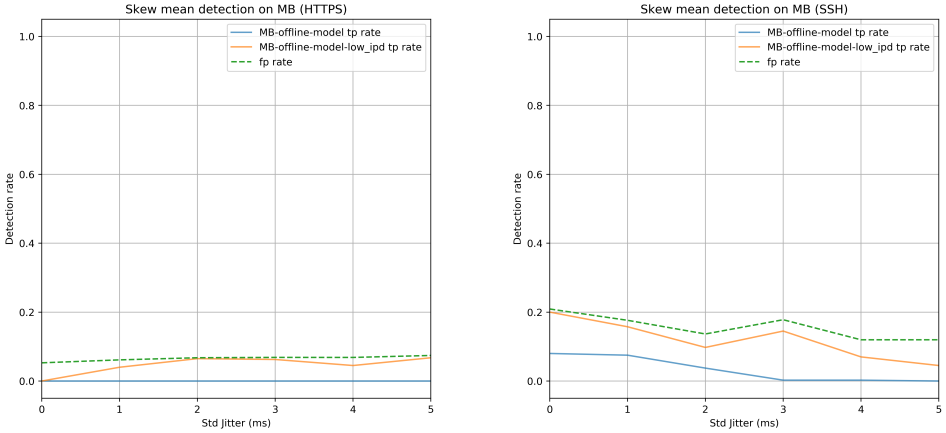


Figure 101: Skew mean detection on MB-CTC

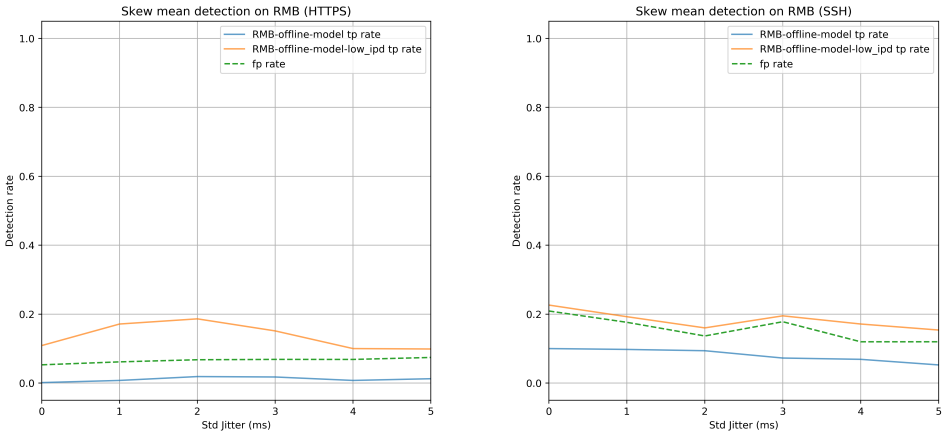


Figure 102: Skew mean detection on RMB-CTC

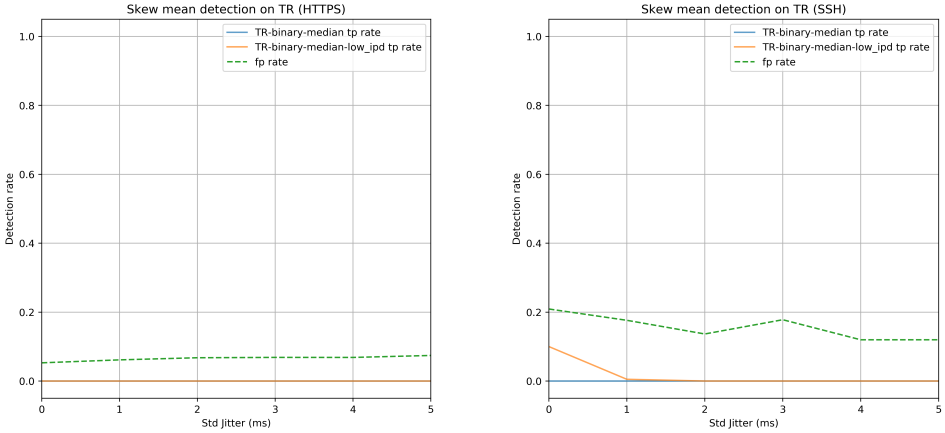


Figure 103: Skew mean detection on TR-CTC

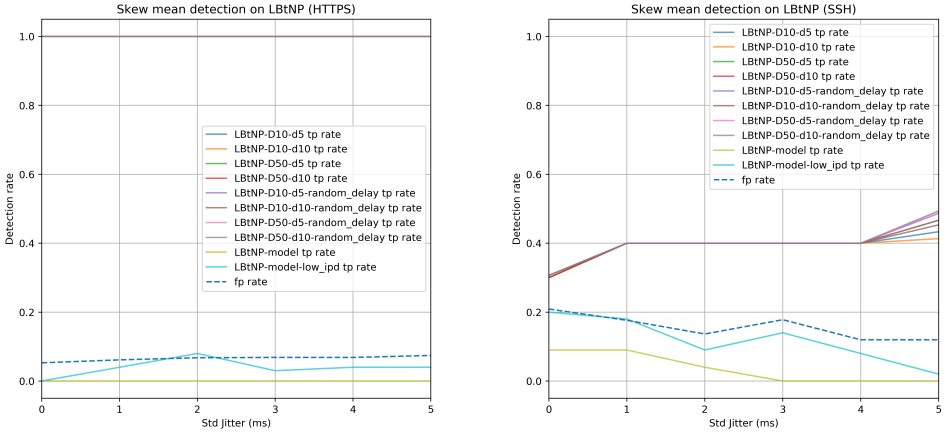


Figure 104: Skew mean detection on LBtNP

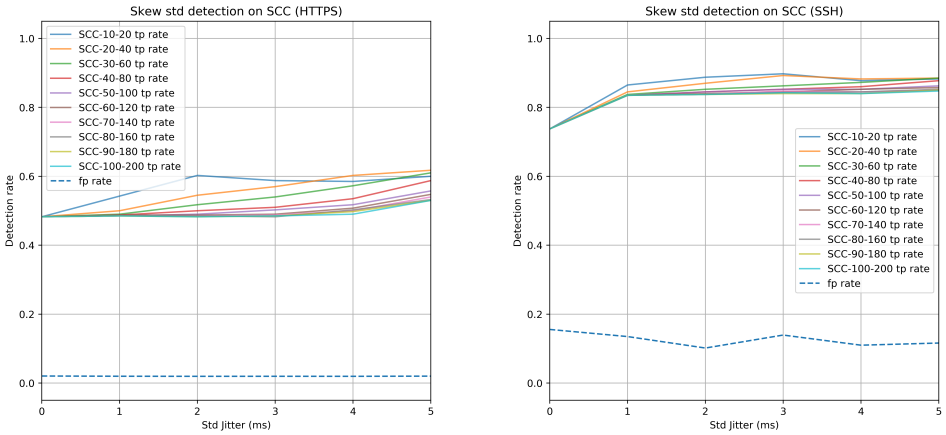


Figure 105: Skew std detection on SCC

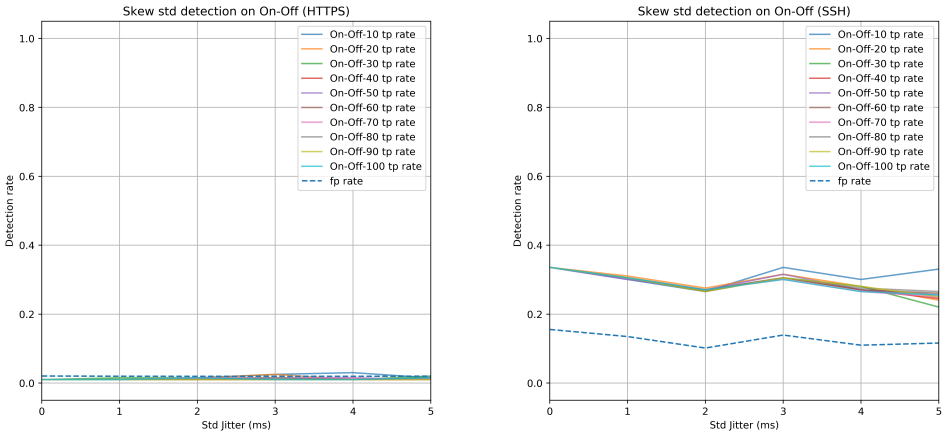


Figure 106: Skew std detection on On-Off

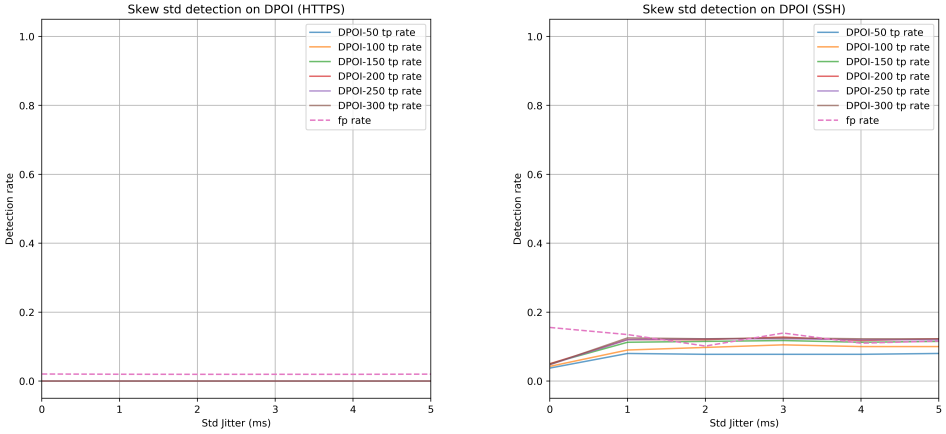


Figure 107: Skew std detection on DPOI

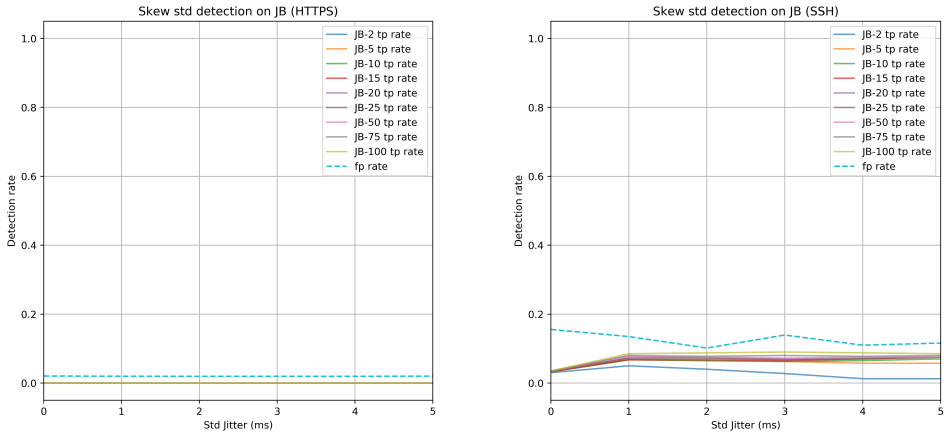


Figure 108: Skew std detection on JB

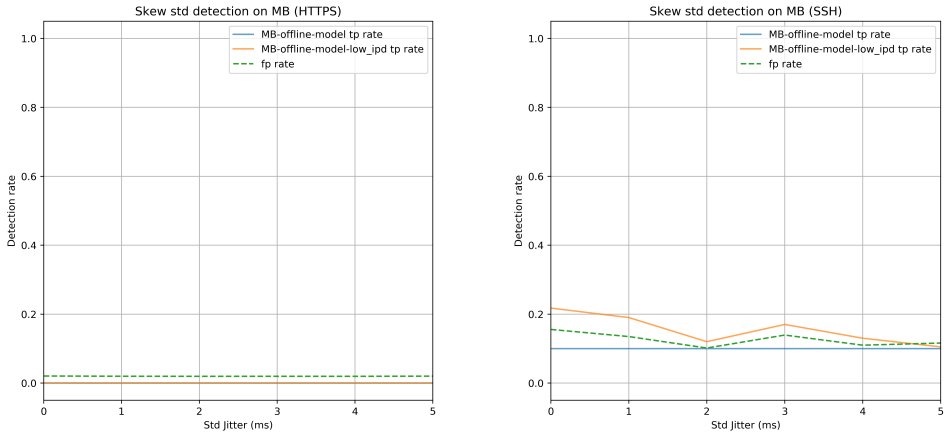


Figure 109: Skew std detection on MB-CTC

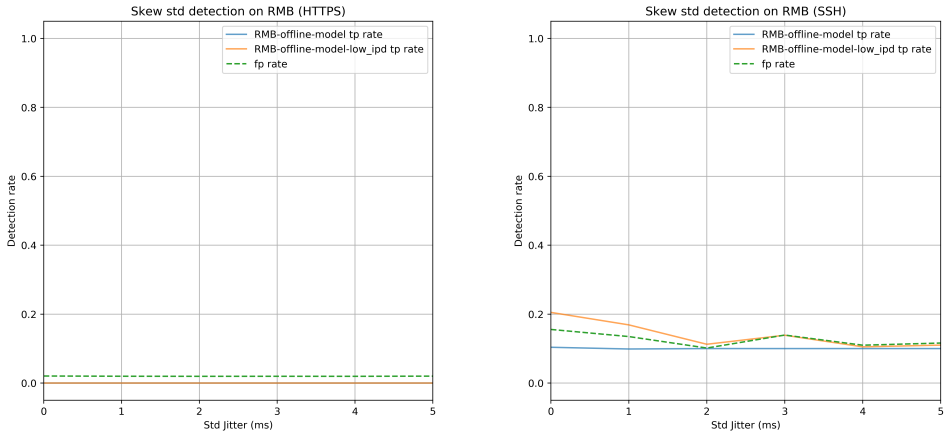


Figure 110: Skew std detection on RMB-CTC

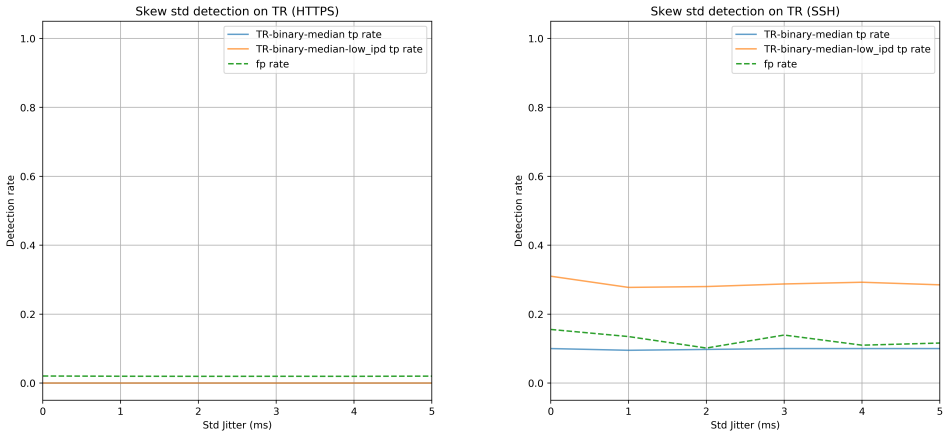


Figure 111: Skew std detection on TR-CTC

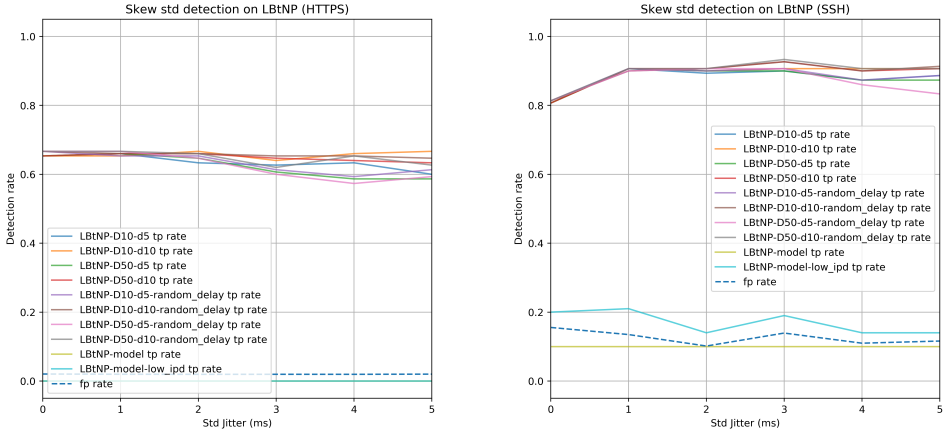


Figure 112: Skew std detection on LBtNP

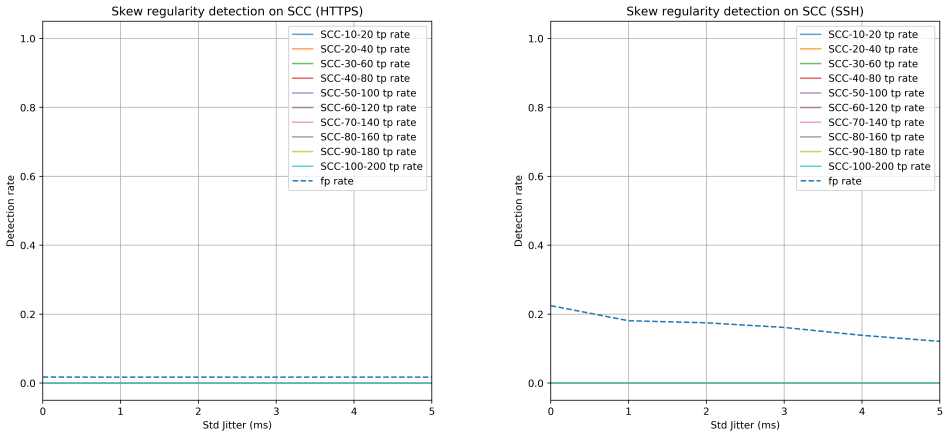


Figure 113: Skew regularity detection on SCC

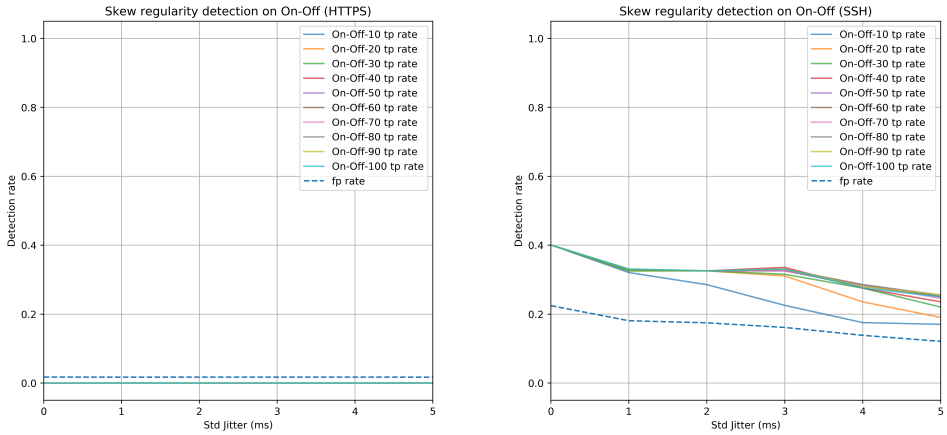


Figure 114: Skew regularity detection on On-Off

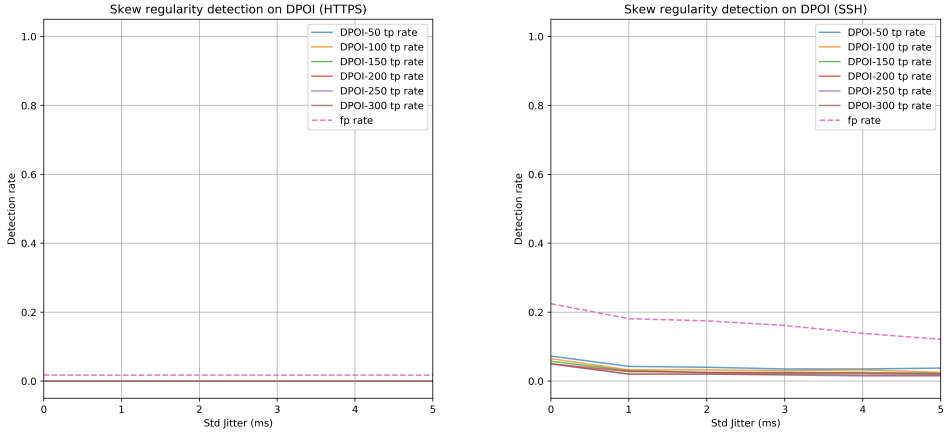


Figure 115: Skew regularity detection on DPOI

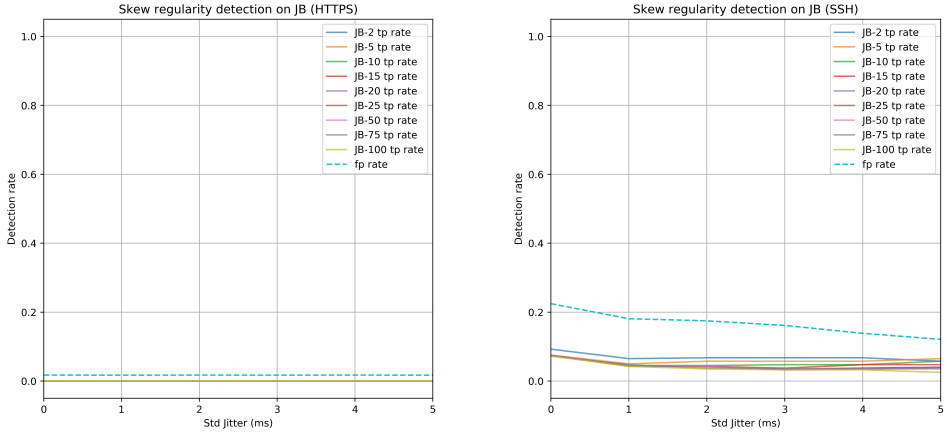


Figure 116: Skew regularity detection on JB

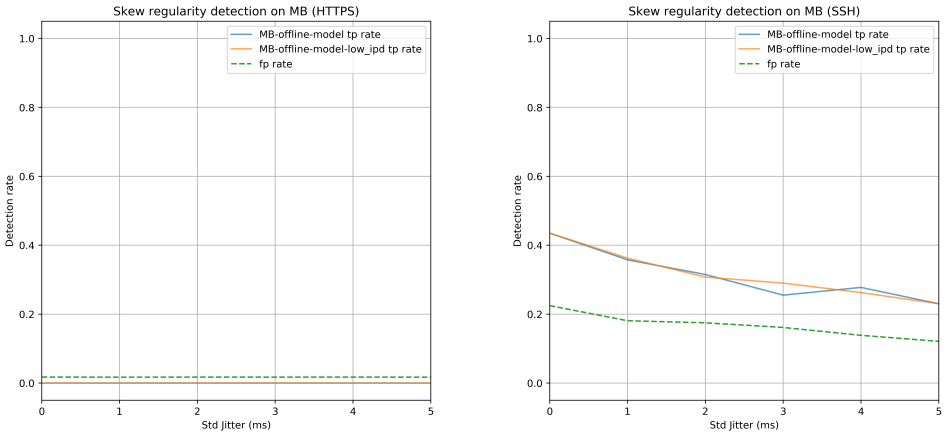


Figure 117: Skew regularity detection on MB-CTC

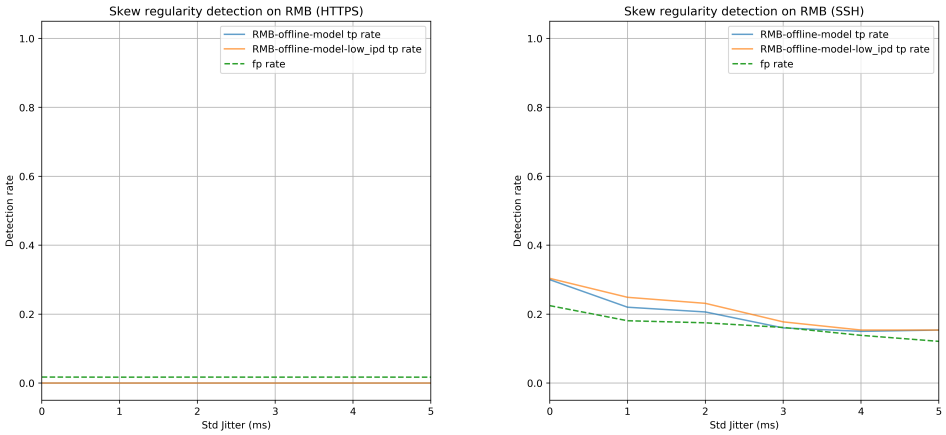


Figure 118: Skew regularity detection on RMB-CTC

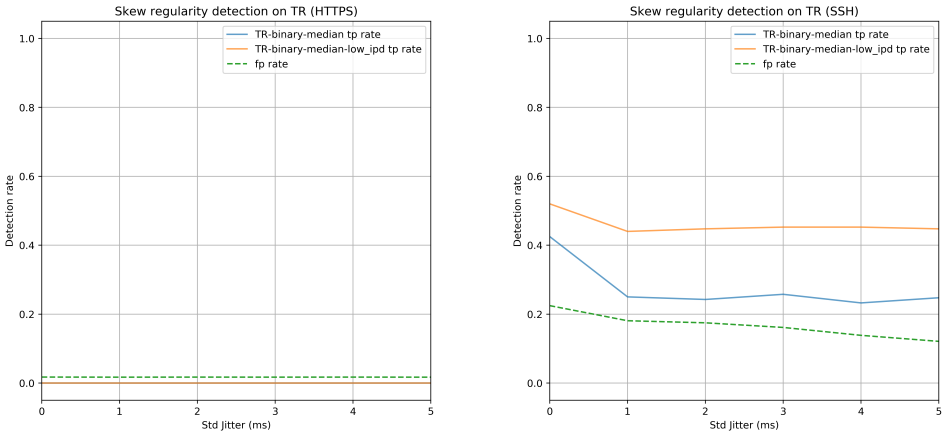


Figure 119: Skew regularity detection on TR-CTC

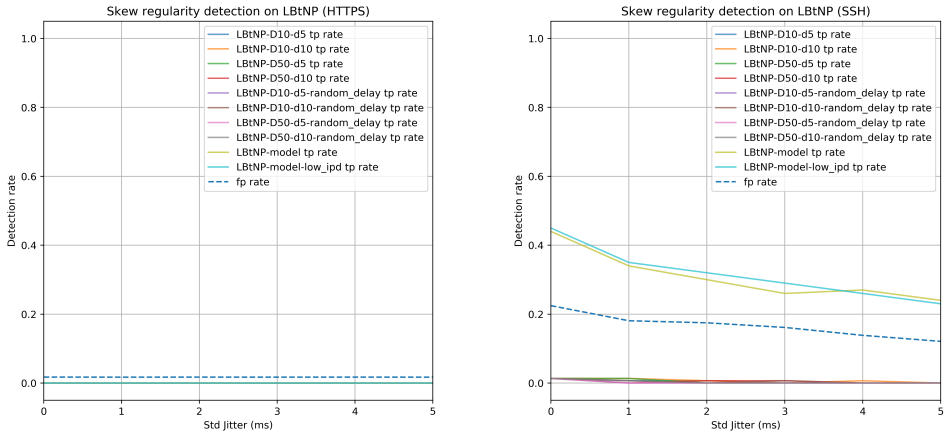


Figure 120: Skew regularity detection on LBtNP

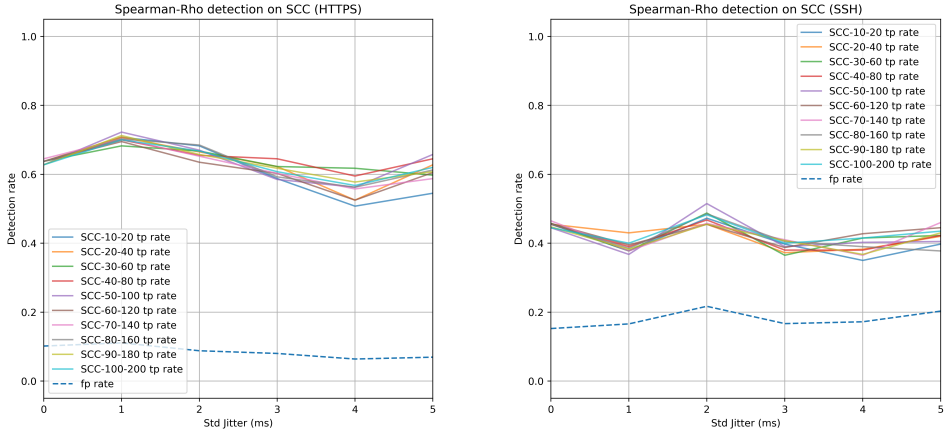


Figure 121: Spearman-Rho detection on SCC

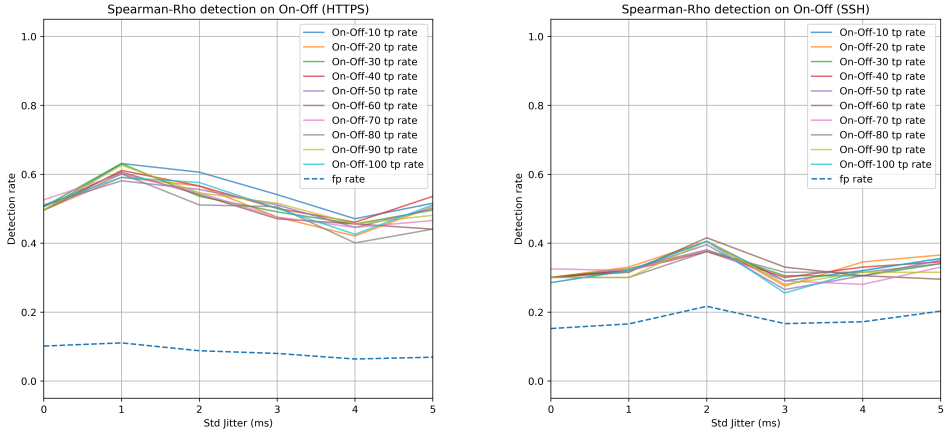


Figure 122: Spearman-Rho detection on On-Off

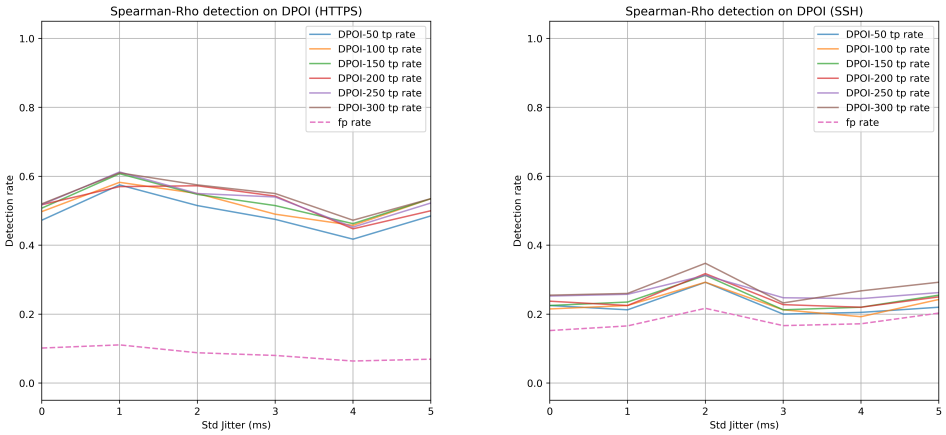


Figure 123: Spearman-Rho detection on DPOI

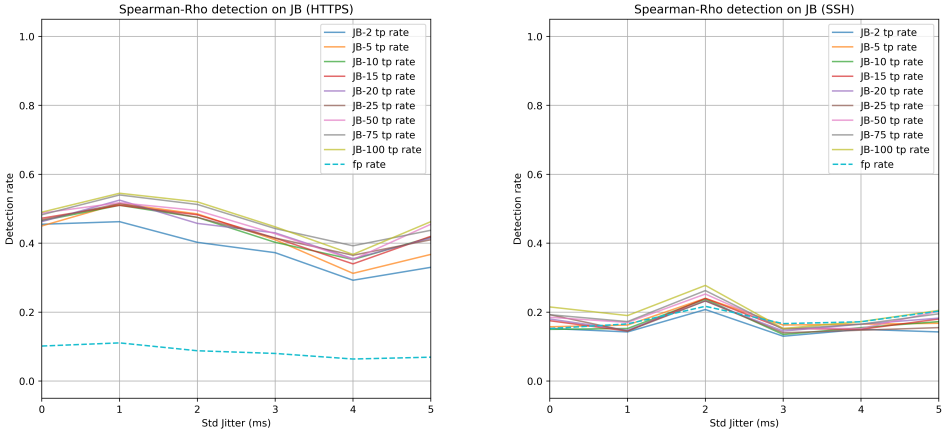


Figure 124: Spearman-Rho detection on JB

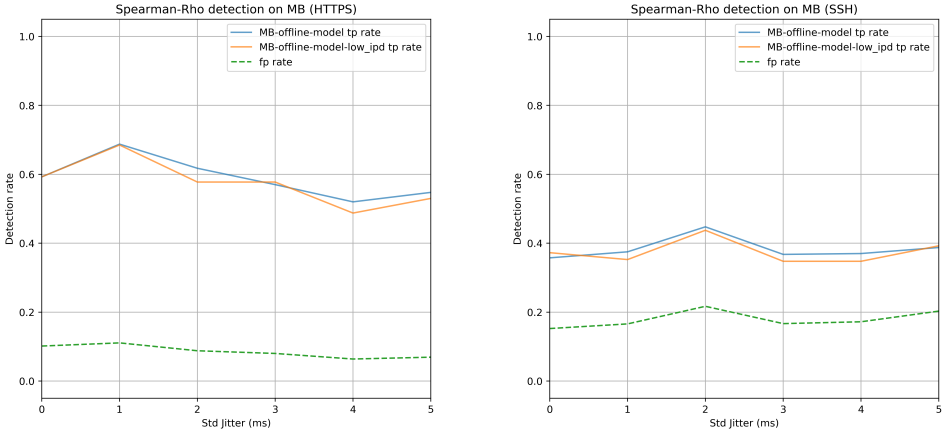


Figure 125: Spearman-Rho detection on MB-CTC

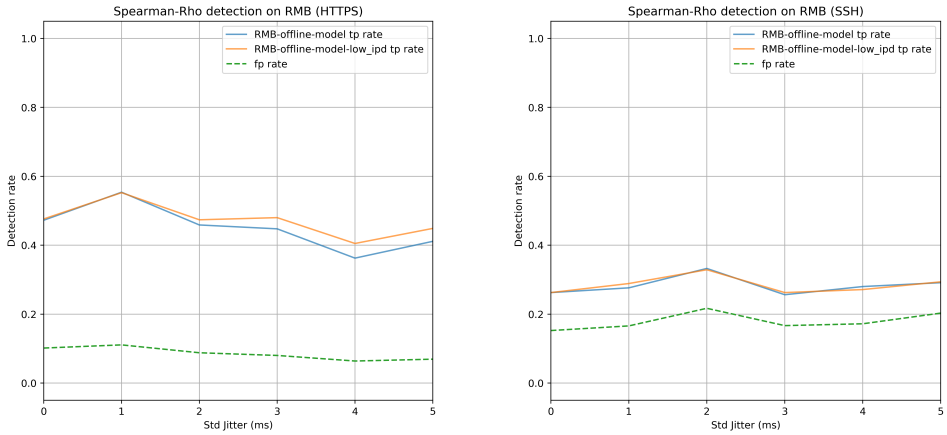


Figure 126: Spearman-Rho detection on RMB-CTC

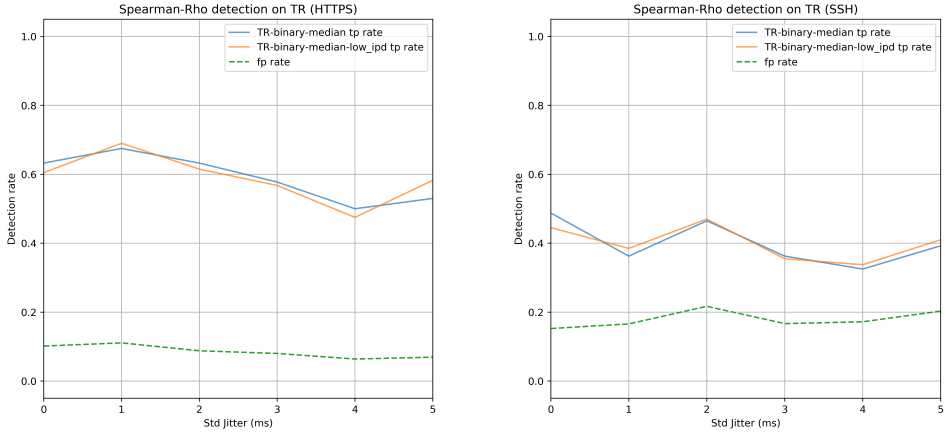


Figure 127: Spearman-Rho detection on TR-CTC

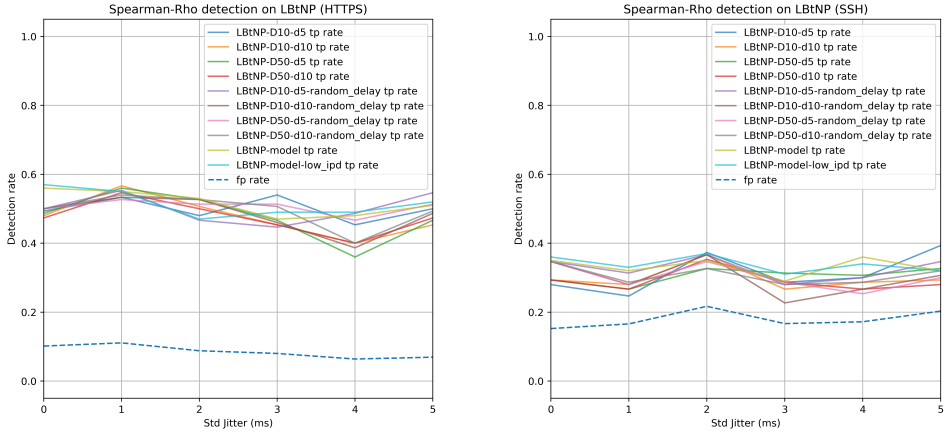


Figure 128: Spearman-Rho detection on LBtNP

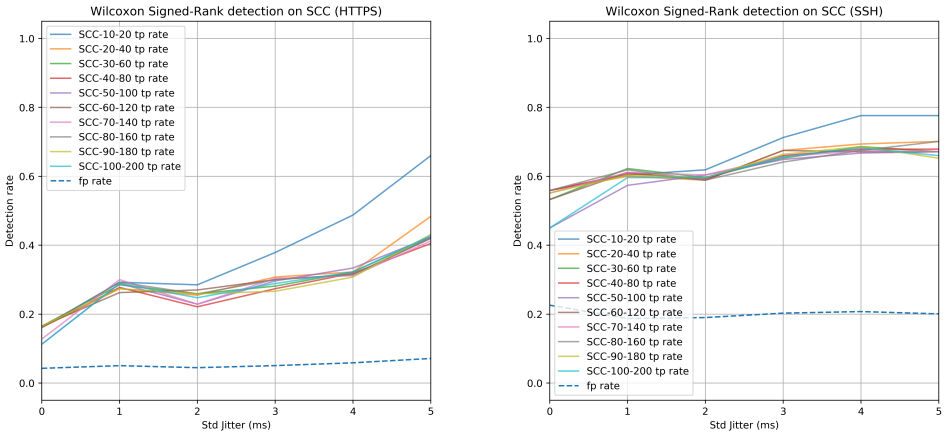


Figure 129: Wilcoxon Signed-Rank detection on SCC

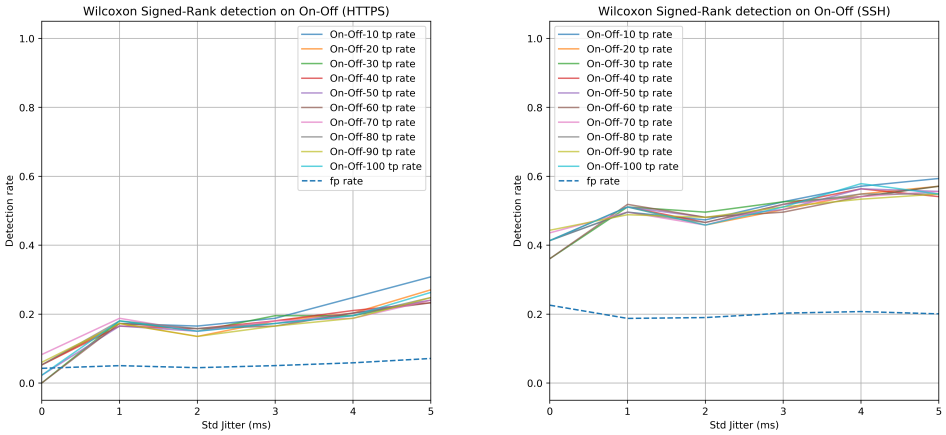


Figure 130: Wilcoxon Signed-Rank detection on On-Off

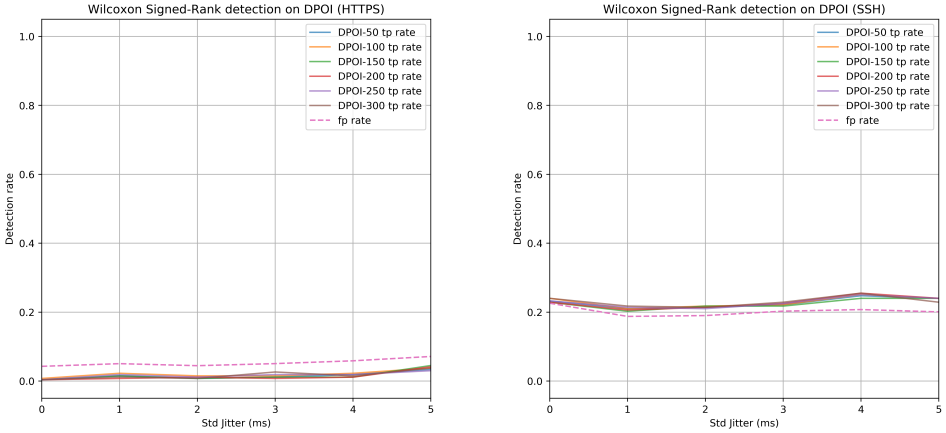


Figure 131: Wilcoxon Signed-Rank detection on DPOI

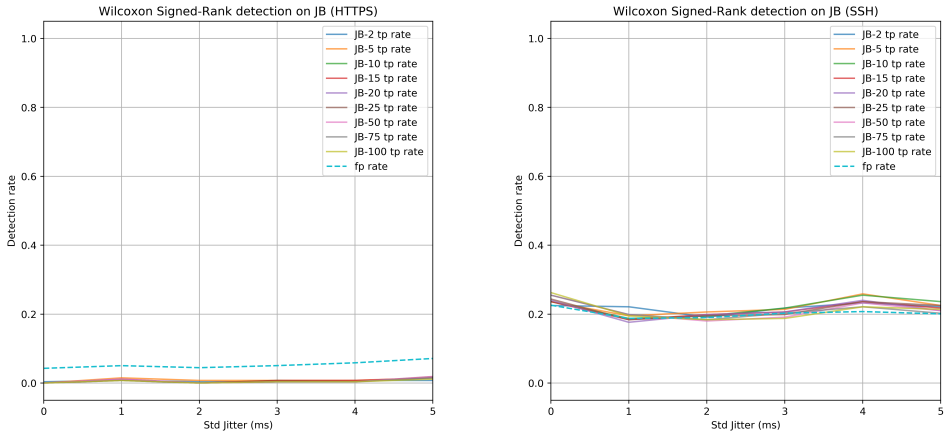


Figure 132: Wilcoxon Signed-Rank detection on JB

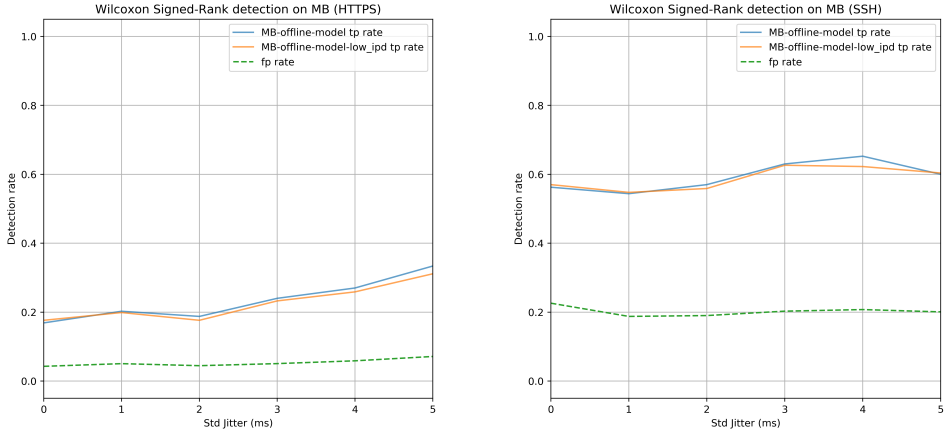


Figure 133: Wilcoxon Signed-Rank detection on MB-CTC

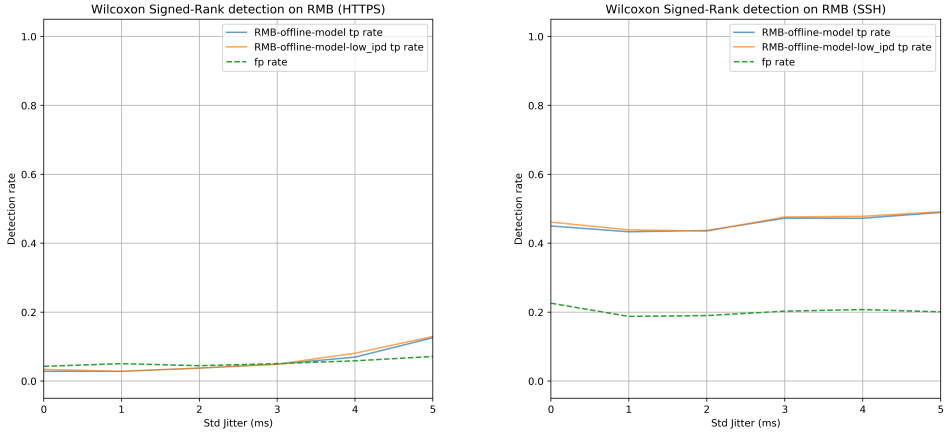


Figure 134: Wilcoxon Signed-Rank detection on RMB-CTC

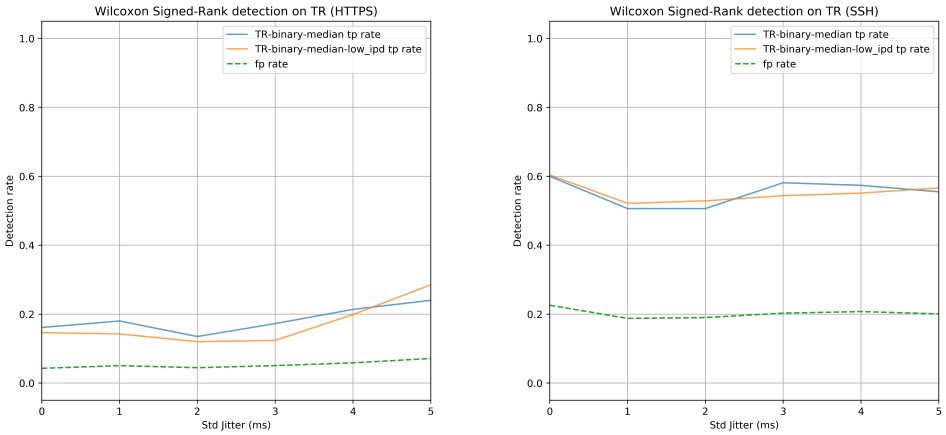


Figure 135: Wilcoxon Signed-Rank detection on TR-CTC

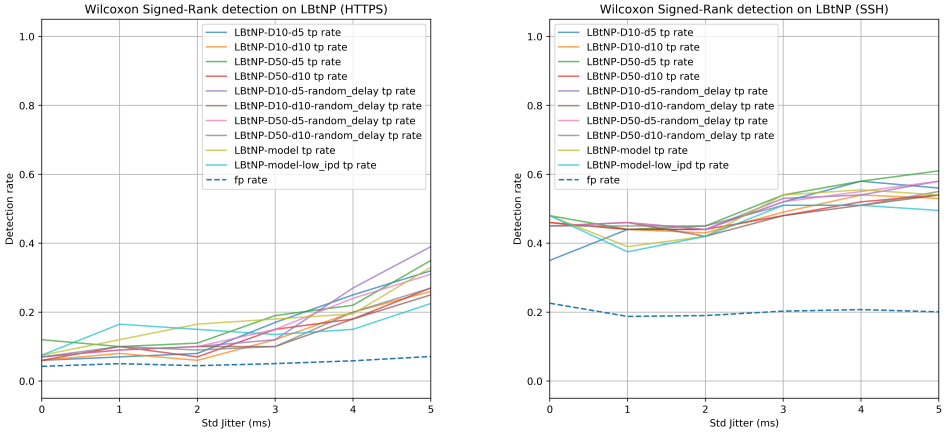


Figure 136: Wilcoxon Signed-Rank detection on LBtNP

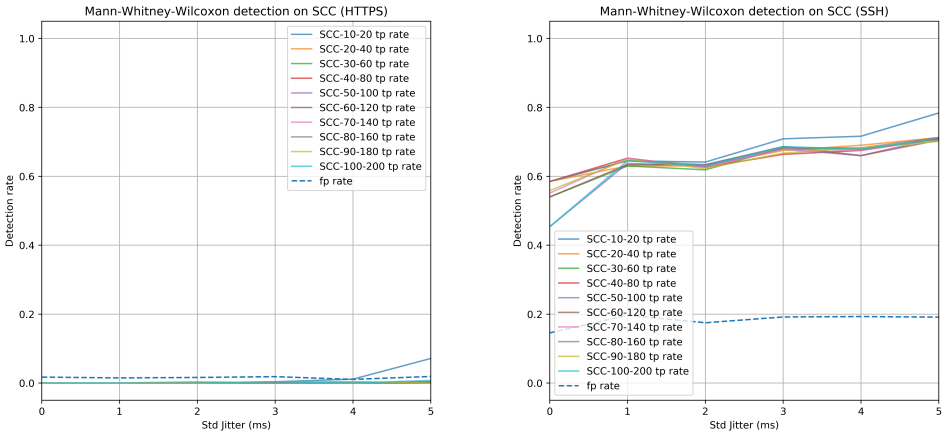


Figure 137: Mann-Whitney-Wilcoxon detection on SCC

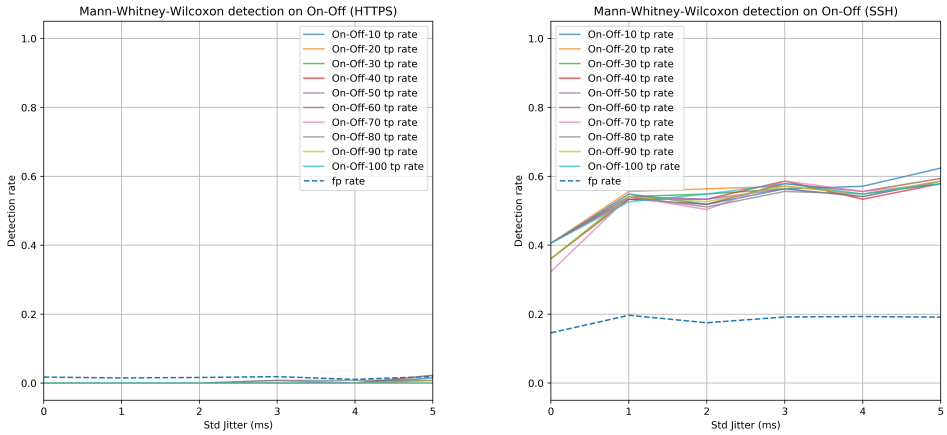


Figure 138: Mann-Whitney-Wilcoxon detection on On-Off

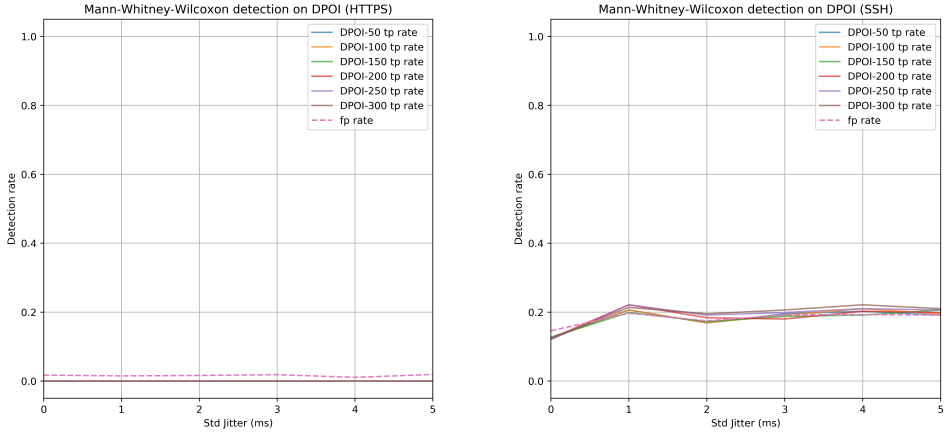


Figure 139: Mann-Whitney-Wilcoxon detection on DPOI

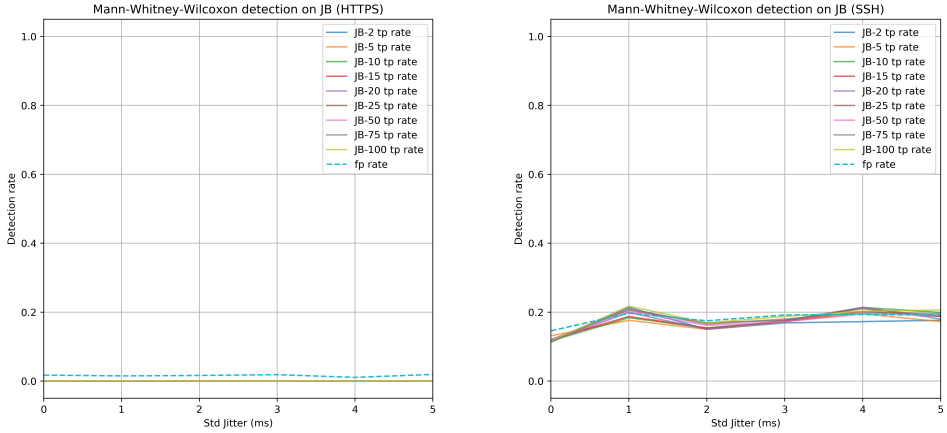


Figure 140: Mann-Whitney-Wilcoxon detection on JB

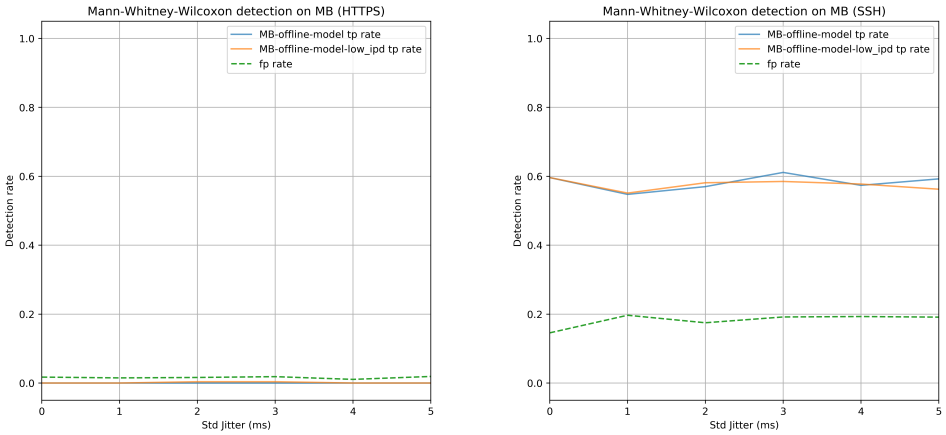


Figure 141: Mann-Whitney-Wilcoxon detection on MB-CTC

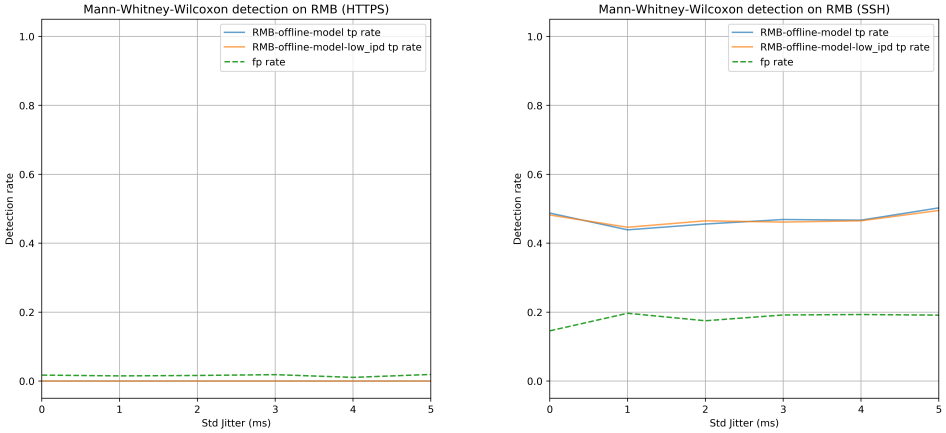


Figure 142: Mann-Whitney-Wilcoxon detection on RMB-CTC

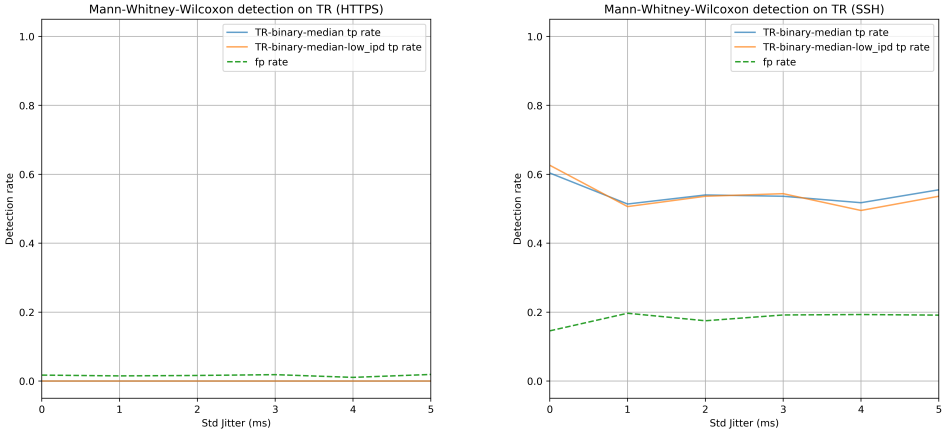


Figure 143: Mann-Whitney-Wilcoxon detection on TR-CTC

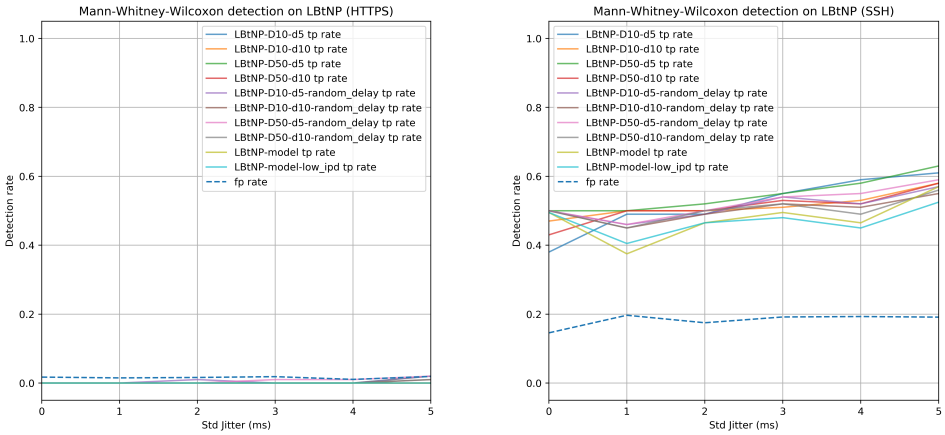


Figure 144: Mann-Whitney-Wilcoxon detection on LBtNP