NETWORK INFORMATION HIDING

CH. 7-A: BASIC NETWORK-LEVEL COUNTERMEASURES

Prof. Dr. Steffen Wendzel
Worms University of Applied Sciences

https://www.wendzel.de (EN) | https://www.hs-worms.de/wendzel/ (DE) @cdp_xe (Twitter)

Online Class: https://github.com/cdpxe/Network-Covert-Channels-A-University-level-Course/

But there are detection methods, right?

- Too many possibilities for realizing network steganography.
  - We need **more** countermeasures!

- Even small FPR can be problematic, if 10 Gbit/s needs to be scanned.
  - Further read (but digital media stego): [1] (100 million new images/photography uploaded and shared every day on social media; even small FPR would render detection impractical if all these images would be analyzed by hand!)
  - We also need **better** countermeasures.

[1] M. Steinebach, A. Ester, H. Liu: Channel Steganalysis, in Proc. ARES'18 (CUING Workshop), ACM, 2018.

Covert Channel Countermeasures

Prevention/Elimination

Limitation

Detection

Several methods exist …

See our book “Information Hiding in Communication Networks” [1], chapter 8, for an overview.

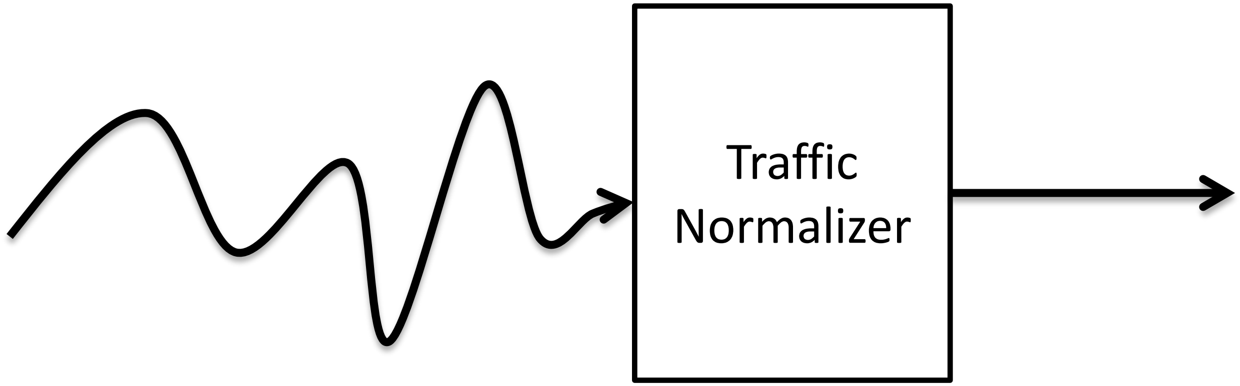
- We will consider only few:
  1. **Elimination**:
     1. Example: **Traffic normalization** to **eliminate** various storage channels
  2. **Detection**:
     1. Example 1: Berk et al. (Inter-packet Times Pattern)
     2. Example 2: Cabuk et al. **epsilon-similarity** (Inter-packet Times Pattern)
     3. Example 3: Cabuk et al. **compressibility score** (Inter-packet Times Pattern)
     4. How to create countermeasures? **Countermeasure variation**.
  3. **Limitation: showing some sophisticated methods:**
     1. Example 1: **Protocol Channel-aware Active Warden** (PCAW) to **limit** protocol switching covert channels/NEL-phase
     2. Example 2: **Dynamic Warden** to **limit** the NEL-phase:

[1] W. Mazurczyk, S. Wendzel, S. Zander et al.: Information Hiding in Communication Networks, WILEY-IEEE, 2016.

Traffic Normalization

- Also known as packet scrubbing, usually part of firewalls and NIPS today, e.g. in OpenBSD pf and Snort.
- In NIH terminology, it is a form of an active warden

- In a nutshell: traffic is modified so that it becomes “normal”, e.g. reserved bits are cleared, some header fields are set to standard values.
  - usually rule-based



How Normalization Works

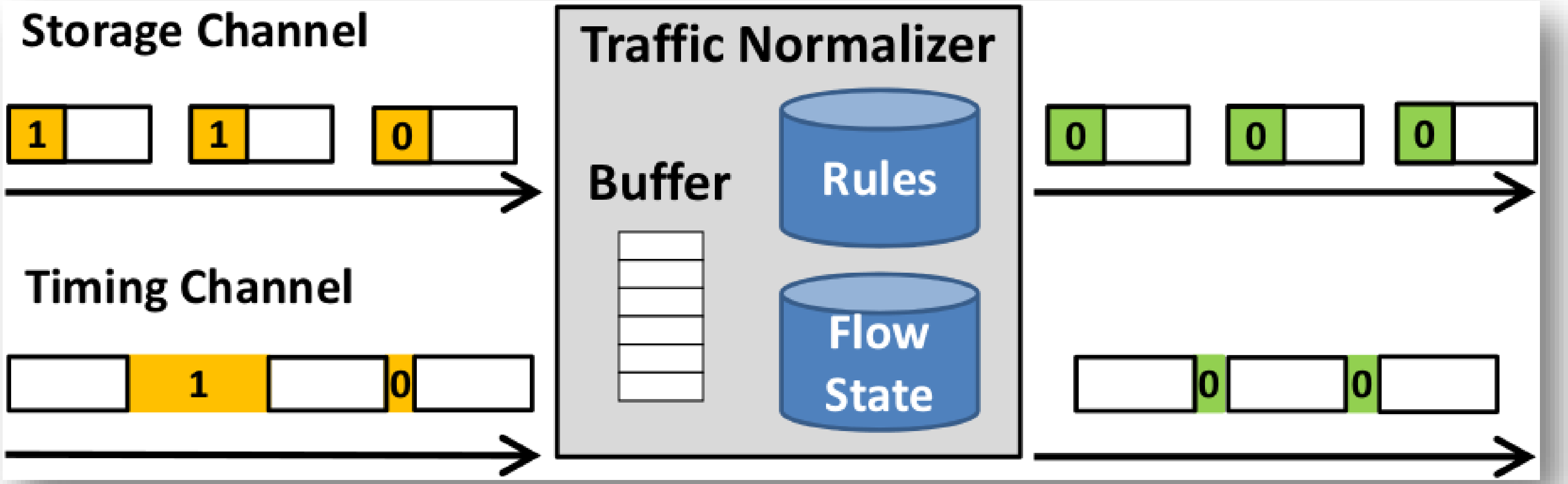


Fig.: [1]

[1] W. Mazurczyk, S. Wendzel, S. Zander et al.: Information Hiding in Communication Networks, WILEY-IEEE, 2016, Chapter 8.

Hochschule Worms University of Applied Sciences

The Problem

Examples for side effects: table at the side [1]

Table 8.2: Well known techniques to normalize IP, UDP and TCP header fields and their possible side effects

Header Field	Normalization Method	Side Effects
IP DF and More Fragments bit, Fragment Offset	Set to zero if packet is below known Maximum Transfer Unit (MTU)	None, assuming packet is not fragmented
IPv4 ToS / Diffserv / ECN, IPv6 Flow Label	Set bits to zero if features unused	None, if bits really not used
IPv4 Time-to-Live, IPv6 hop limit	Set to a fixed value larger than longest path necessary	Higher bandwidth consumption if routing loops
IP source	Drop packet if private, localhost, broadcast address	Malformed packets are dropped
IP destination	Drop packet if destination private or non-existent	Some packets are dropped
IP ID field	Rewrite/scramble IP ID fields	May impact diagnostics relying on increasing IDs
IPv4 Options	Remove all options	May impact functionality, but IPv4 options are rarely used
IPv6 Options	Many normalization techniques proposed in [41]	See [41]
Fragmented IP packets	Reassemble and refragment if necessary	None
TCP and other timestamps	Randomize low order bits of timestamps	None, if noise introduced is low
IP, UDP, TCP packet length	If incorrect discard or trim packets	Malformed packets are dropped
IP, UDP, TCP header length	Drop packet with header length smaller than minimum	Malformed packets are dropped
IP, UDP, TCP checksums	Drop packet if incorrect	Malformed packets are dropped
Padding in header options	Zero padding bits	None
TCP Sequence and Ack numbers	Rewrite initial and following sequence numbers and convert Ack numbers back to original sender number space	None

[1] W. Mazurczyk, S. Wendzel, S. Zander et al.: Information Hiding in Communication Networks, Ch. 8, Wiley-IEEE, 2016.

Hochschule Worms
University of Applied Sciences

Inconsistent TCP-Retransmissions

- Handley et al. [1]:
  - How to handle overlapping TCP segments as such caused by re-transmissions, especially if their payload differs?

  - Example (based on [1]):

```
seq:1, TTL:10, payload=n
seq:1, TTL:12, payload=y
seq:2, TTL:11, payload=o
seq:2, TTL:12, payload=e
seq:3, TTL:10, payload=!
seq:3, TTL:11, payload=s
```

    - We could receive different messages: „`yes`“, „`no!`“, „`yo!`“, …

  - Depending on the TTL: Which segments will reach the receiver?
  - What are the potential consequences of the different scenarios?
  - We need to cache all the data and evaluate all possibilities in the TN.

[1] M. Handley et al.: Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics, Proc. Usenix Symp. 2001.
https://www.usenix.org/legacy/events/sec01/full_papers/handley/handley.pdf

Cold Start

- Handley et al. [1]:

  [The design of a TN] *"can prove vulnerable to incorrect analysis during a* ***cold start****. That is, when the analyzer first begins to run, it is confronted with traffic from already-established connections* for which *the analyzer lacks knowledge of the connection characteristics negotiated when the connections were established.*

  For example, the TCP scrubber [8] requires a connection to go through the normal start-up handshake. However, if a valid connection is in progress, and the scrubber restarts or otherwise loses state, then it will terminate any connections in progress at the time of the cold start, since to its analysis their traffic exchanges appear to violate the protocol semantics that require each newly seen connection to begin with a start-up handshake."

[1] M. Handley et al.: Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics, Proc. Usenix Symp. 2001.
https://www.usenix.org/legacy/events/sec01/full_papers/handley/handley.pdf

Stateholding Problem

- Handley et al. [1]:

  „A NIDS system must hold state in order to understand the context of incoming information. One form of attack on a NIDS is a stateholding attack, whereby the attacker creates traffic that will cause the NIDS to instantiate state (…). If this state exceeds the NIDS's ability to cope, the attacker may well be able to launch an attack that passes undetected.

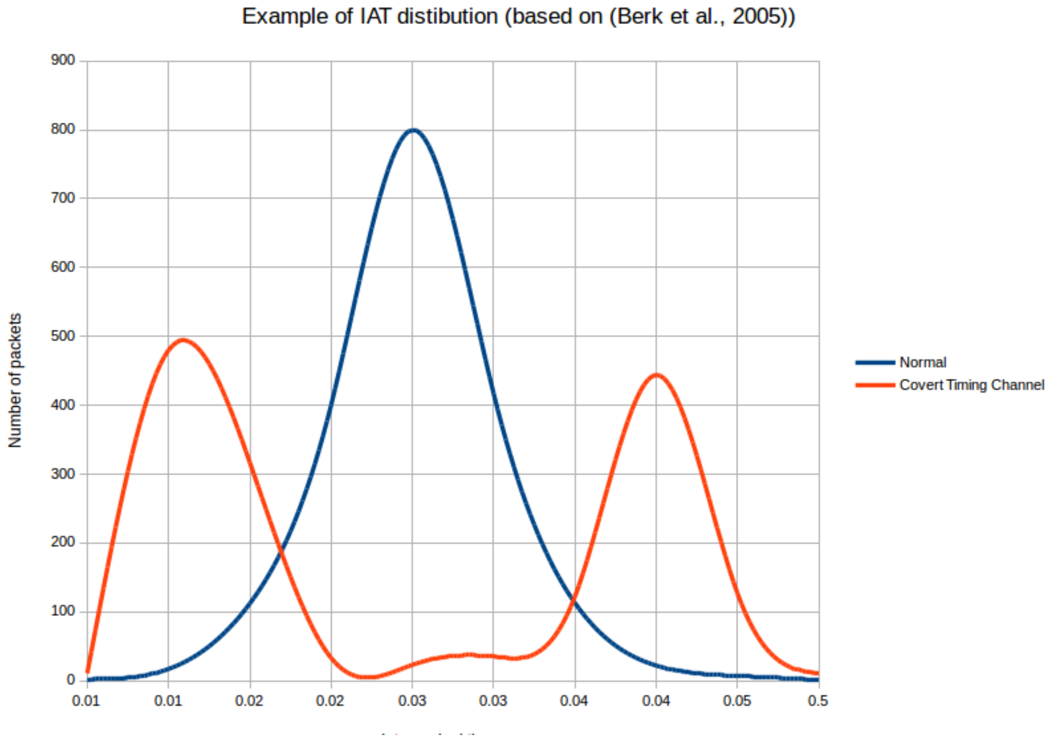
  […]

  An attacker can thus cause the normalizer to use up memory by sending many fragments of packets without ever sending enough to complete a packet."

[1] M. Handley et al.: Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics, Proc. Usenix Symp. 2001.
https://www.usenix.org/legacy/events/sec01/full_papers/handley/handley.pdf

Inter-packet Times Pattern: Detection by Berk et al.

- (Berk et al., 2005) state that IPGs of non-covert traffic are distributed in a way that most of the measured IPGs are close to an average IPG value X.
- Inter-arrival time-based covert channels, however, signal hidden information using at least two different inter-arrival time encoded symbols, resulting in at least two, instead of one `peak‘ of IPG values:
- Procedure:
  - Record all IPGs
  - C_μ: # packets with avg. IAT value
  - C_{max}: highest number of packets with same IAT
  - $P_{CovChan} = 1 - \frac{C_\mu}{C_{max}}$

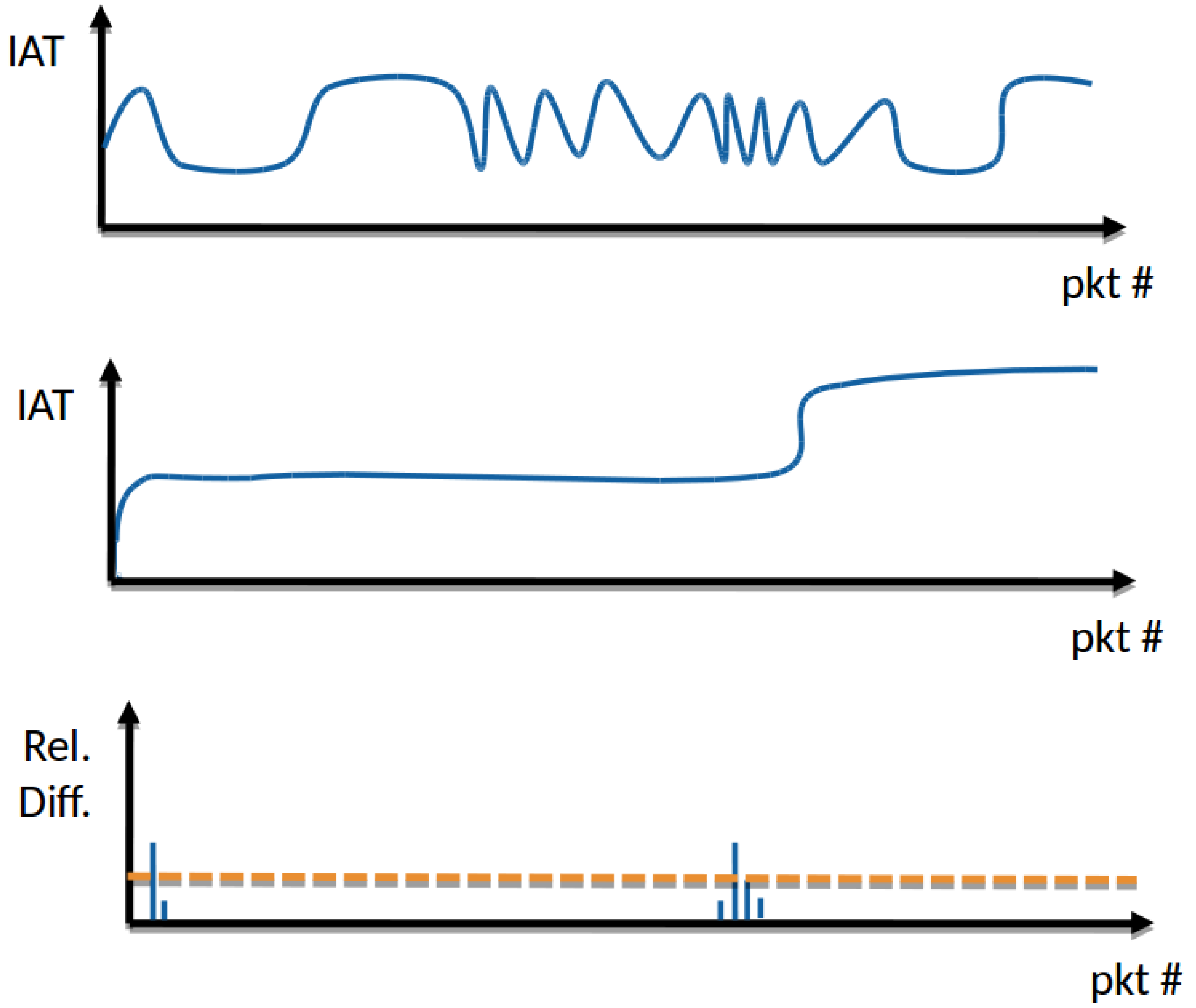


[1] V. Berk, A. Giani, G. Cybenko: Detection of Covert Channel Encoding in Network Packet Delays, Technical Report, Dep. Comp. Sc., Dartmouth College, Hanover, NH, 2005.

Inter-packet Times Pattern: Detection by Cabuk et al.: **ε-similarity**

Introduced by Cabuk et al. in [1].

1. Record all inter-packet gaps of a flow.
2. Sort all inter-packet times of a flow.
3. For consecutive values T_1 and T_2: calculate relative difference $\lambda_i = \frac{|T_{i+1} - T_i|}{T_i}$.
4. Calculate the percentage of λ values of a given flow that are below the threshold ε.



[1] S. Cabuk et al.: IP Covert Channel Detection, in: Transactions on Information and System Security (TISSEC), ACM, 2009.

Inter-packet Times Pattern: Detection by Cabuk et al.: **Compressibility Score**

Introduced by Cabuk et al. [1]:

In a nutshell:

1. Record all inter-packet times of a flow $\Delta_{t_1}, \ldots, \Delta_{t_n}$.
2. Encode the inter-packet times in an ASCII string S, e.g. “A20A20A19B30B29A20...”.
3. Compress S with a compressor $\mathfrak{J}$ (e.g. *gzip*): $C = \mathfrak{J}(S)$.
4. Use $\kappa = \frac{|S|}{|C|}$ as an indicator for the presence of a covert channel.

[1] S. Cabuk et al.: IP Covert Channel Detection, in: Transactions on Information and System Security (TISSEC), ACM, 2009.

2015-overview of potential countermeasures in combination with patterns [1]

Table III. Application of Covert Channel Countermeasures to Patterns

	Elimination	Limitation	Detection
Storage Channel Patterns			
P1. Size Modulation			SA/ML
P2. Sequence	TN		SA/ML
P2.a. Position	TN		SA/ML
P2.b. Number of Elements	TN		SA/ML
P3. Add Redundancy	TN		SA/ML
P4. PDU Corruption/Loss	TN		SA/ML
P5. Random Value	TN		SA/ML
P6. Value Modulation		TN (limited), NPRC	SA/ML
P6.a. Case	TN		SA/ML
P6.b. LSB	TN		SA/ML
P7. Reserved/Unused	TN		SA/ML
Timing Channel Patterns			
P8. Interarrival Time		TN (limited), NPRC	SA/ML
P9. Rate		TN (limited), NPRC	SA/ML
P10. PDU Order		TN (limited) NPRC	SA/ML
P11. Retransmission			SA/ML

TN: Traffic Normalization
NPRC: Network Pump and Related Concepts
SA/ML: Statistical Approaches/Machine Learning

[1] S. Wendzel et al.: Pattern-based Survey and Taxonomy for Network Covert Channels, ACM CSUR, Vol. 47(3), 2015.

New Pattern, No Countermeasure?

COUNTERMEASURE VARIATION

Countermeasure Variation [1]

Problem: We lack countermeasures for several of the known patterns.

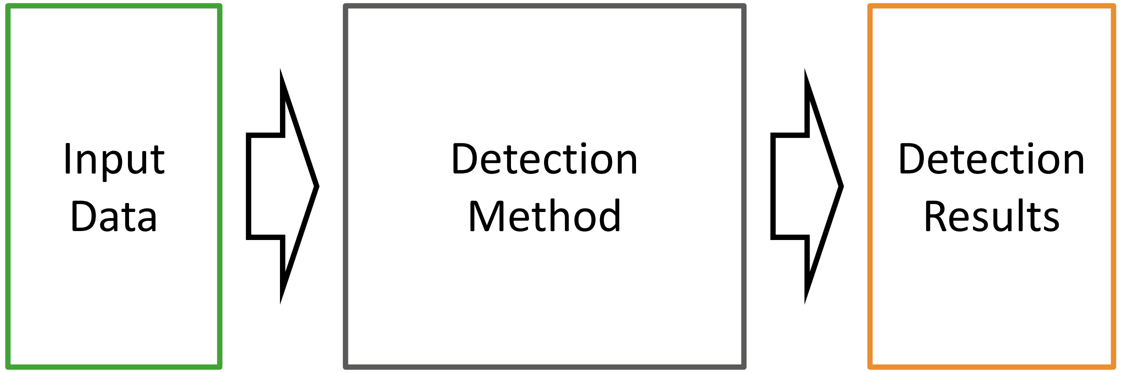
Solution: Introduction of **countermeasure variation**.

Definition. Given the two hiding patterns A and B, with $A \neq B$, a *countermeasure variation* is a pattern-based process in which an existing countermeasure that detects, limits, prevents or audits covert channels of pattern A is modified so that it detects, limits, prevents or audits covert channels of pattern B.
The process of countermeasure variation replaces the input attributes (*features*) used for A with features for B and performs a modification of the inner functioning (e.g., the algorithm) used for A in order to work with the new features for B. The alternation of the inner functioning is kept as small as possible, which provides the contrast to developing entirely new countermeasures. In comparison to simply applying the same countermeasure (e.g. a statistical method) to another covert channel technique, countermeasure variation *i) requires* the modification of the inner functioning and *ii)* focuses on hiding patterns, i.e., it needs to consider features that can be used for multiple covert channels belonging to the same pattern. ■

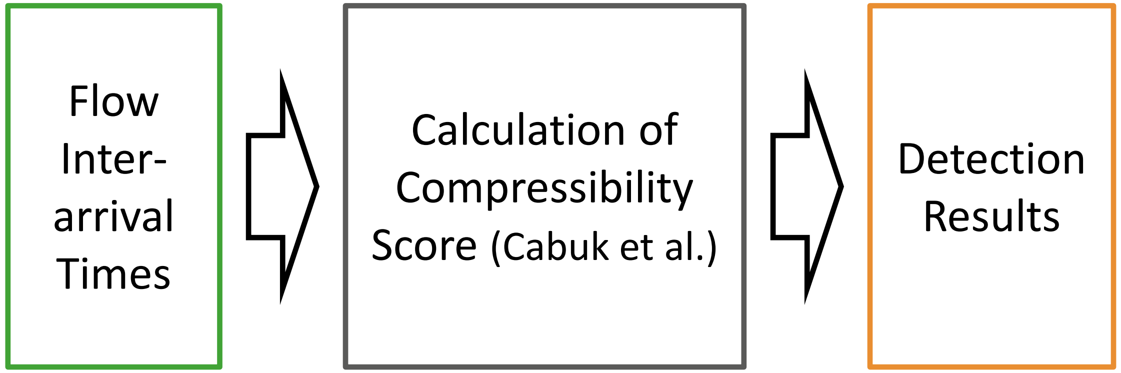
[1] S. Wendzel et al.: Detection of Size Modulation Covert Channels Using Countermeasure Variation, Journal of Universal Computer Science (J.UCS), Vol. 25(11), pp. 1396-1416, 2019.

Countermeasure Variation [1]

Classic covert channel countermeasures look like this:



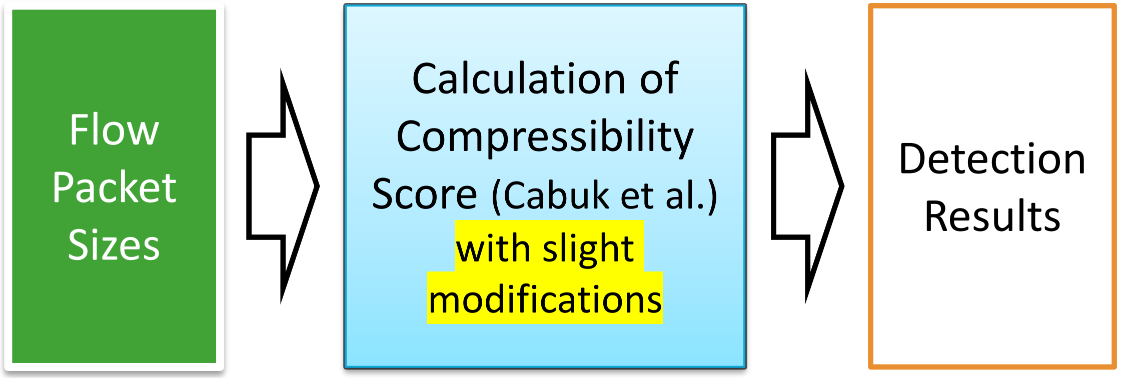
For instance:



[1] S. Wendzel et al.: Detection of Size Modulation Covert Channels Using Countermeasure Variation, Journal of Universal Computer Science (J.UCS), Vol. 25(11), pp. 1396-1416, 2019.

Countermeasure Variation [1]

Countermeasure Variation modifies the input to the detection method and alters the detection method as little as possible.



[1] S. Wendzel et al.: Detection of Size Modulation Covert Channels Using Countermeasure Variation, Journal of Universal Computer Science (J.UCS), Vol. 25(11), pp. 1396-1416, 2019.

Countermeasure Variation

So far, we performed countermeasure variation for
- Compressibility Score
- ε-similarity

Each in combination with the following patterns:
- Size Modulation
- Artificial Re-transmission
- *Sequence Modulation*
- Message Ordering

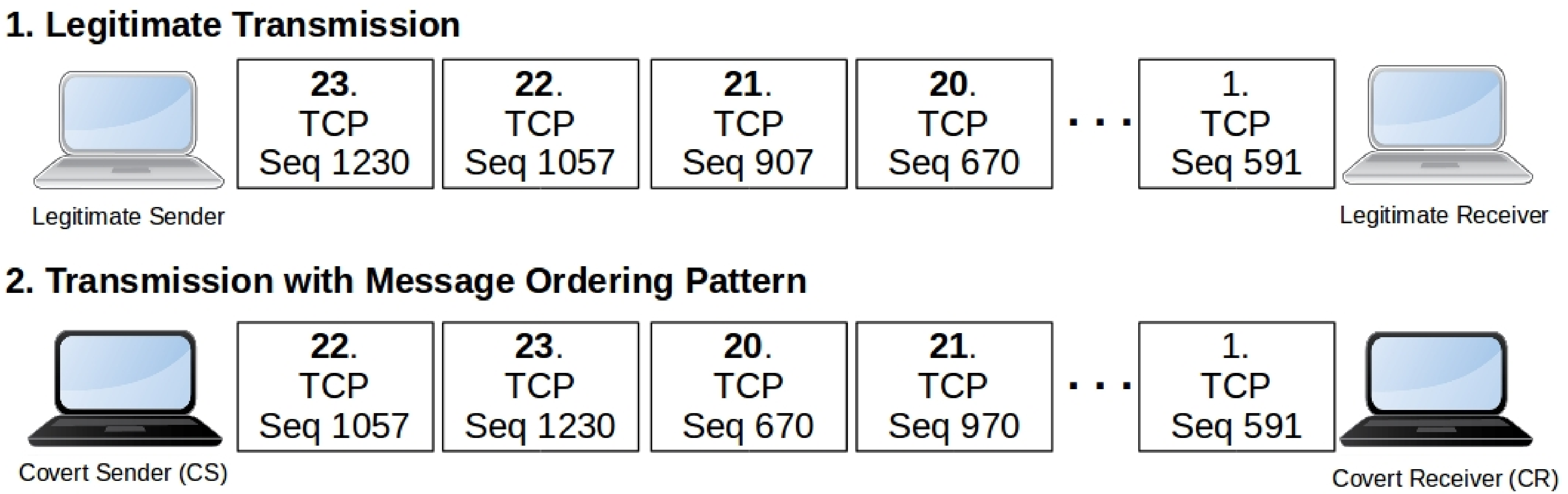


Fig.: S. Wendzel: [Protocol-independent Detection of "Messaging Ordering" Network Covert Channels](), in Proc. CUING, ACM, 2019.

Countermeasure Variation: Compressibility Score for Message Ordering Channels [1]

1. Create Traffic Recording

TCP Seq 140	TCP Seq 160	TCP Seq 120	TCP Seq 100	. . .	TCP Seq 10

2. Enumerate Packets

TCP Segment #37	TCP Segment #38	TCP Segment #36	TCP Segment #35	. . .	TCP Segment #1

3. Calculate Relative Differences

-1	+2	+1	. . .

window size: 200 differences

4. Concatenate Differences in a String

Coding Type #1:
`S = “...12-1“`

5. Compress String

`C = compress(S)`

6. Calculate Compressibility

`K = |C|/|S|`

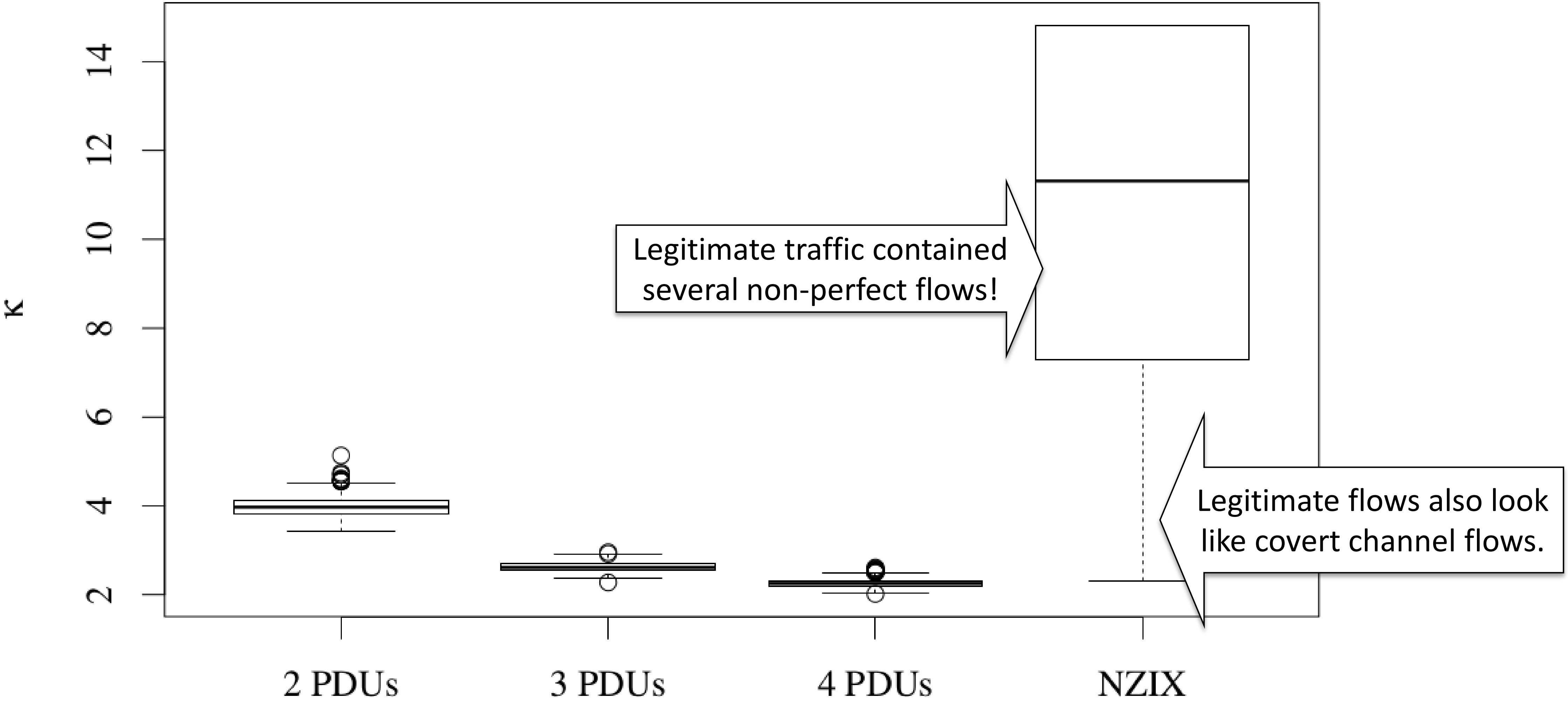
experimented with different string encoding types; best results with
`|diff| || (|diff| % 2 ? ’A’ : ’B’)`

Thus, optimal flow would produce the string “**1A1A1A1A…1A1A1A**” with **|S| = 400**, resulting in **|C| = 27** and thus **κ = 14.81481** (max value).

[1] S. Wendzel: Protocol-independent Detection of “Messaging Ordering” Network Covert Channels, in Proc. CUING, ACM, 2019.

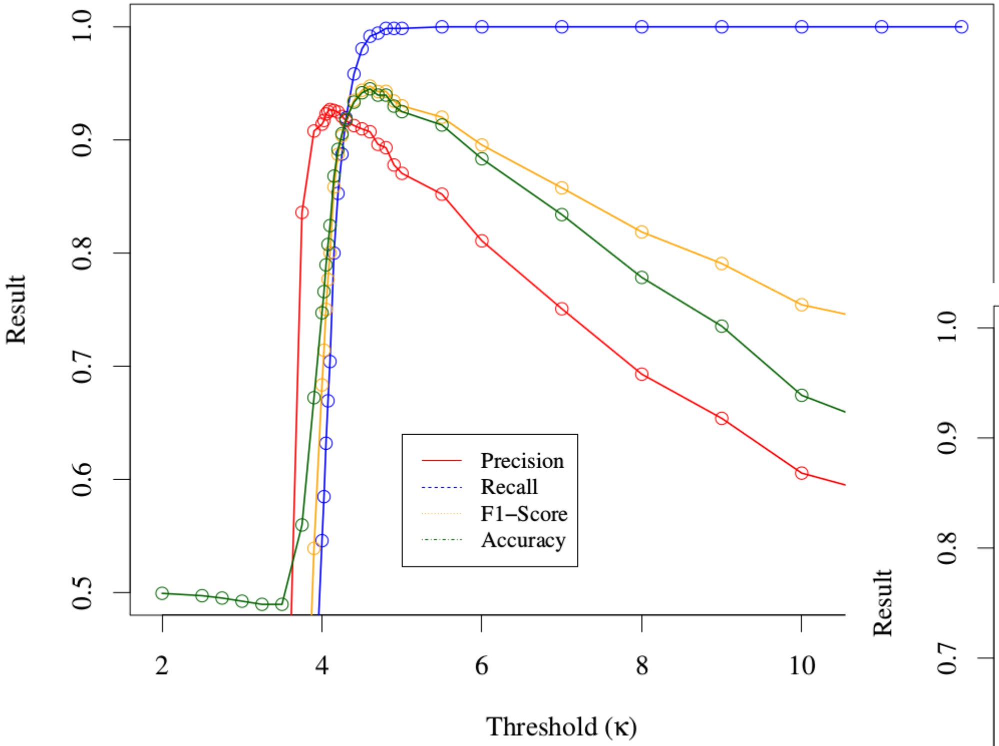
Countermeasure Variation: Compressibility Score for Message Ordering Channels [1]

Kappa values for different types of covert channels and legitimate traffic (NZIX).
PDUs follow a uniform distribution to reflect encrypted content.

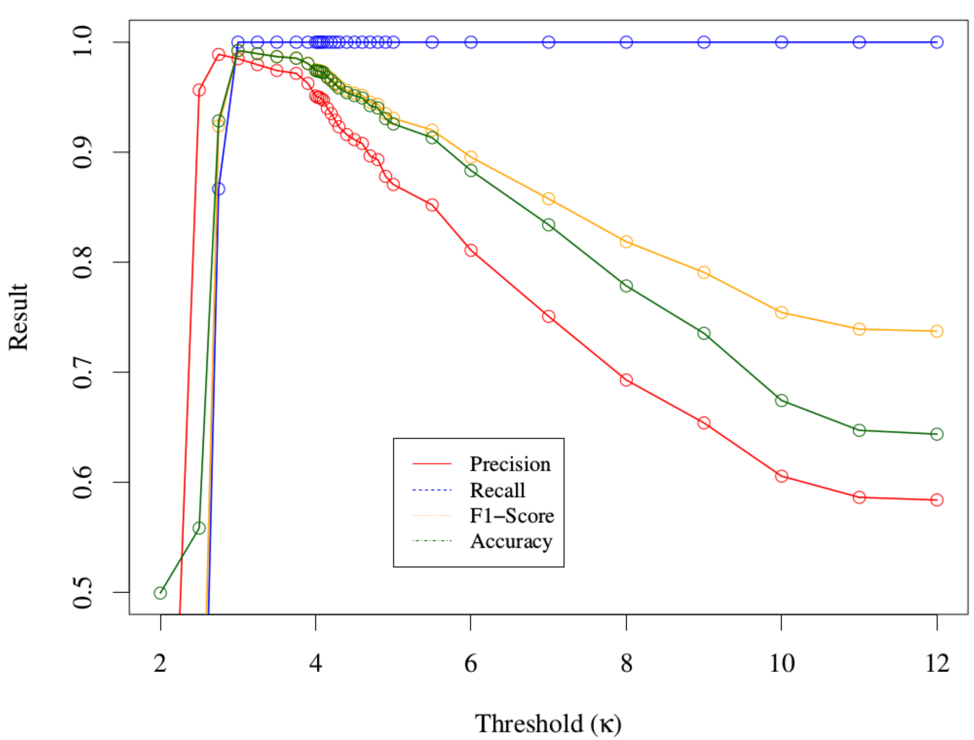


[1] S. Wendzel: Protocol-independent Detection of “Messaging Ordering” Network Covert Channels, in Proc. CUING, ACM, 2019.

Message Ordering Pattern: Detection (Ethernet) [1]



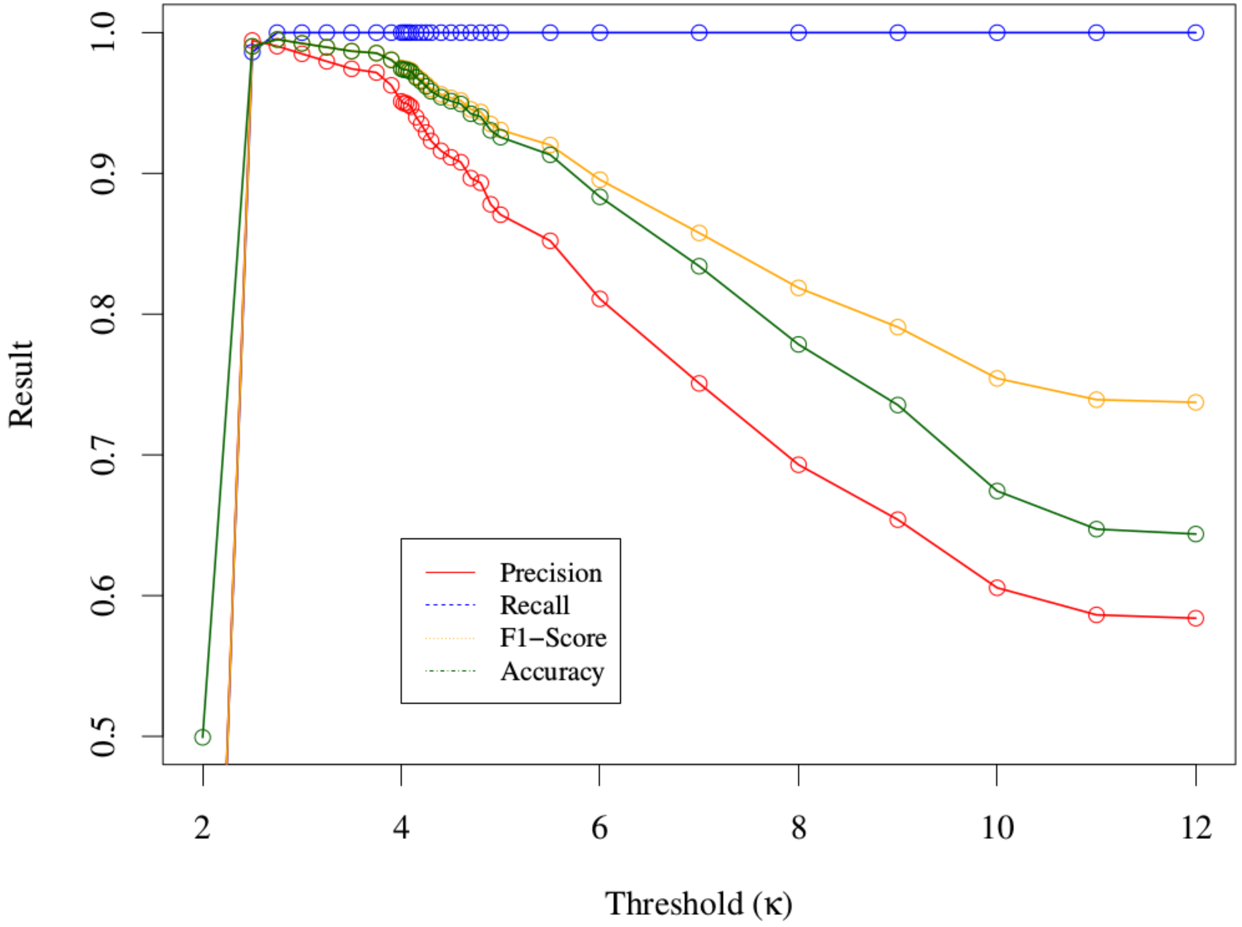
2-PDU channel:
- 94.513% accuracy
- 94.756% F-score



3-PDU channel:
- 99.236% accuracy
- 99.241% F-score

[1] S. Wendzel: Protocol-independent Detection of "Messaging Ordering" Network Covert Channels, in Proc. CUING, ACM, 2019.

Message Ordering Pattern: Detection (Ethernet) [1]



More PDUs pose a higher threat but can be detected better!

4-PDU channel:

99.513% accuracy

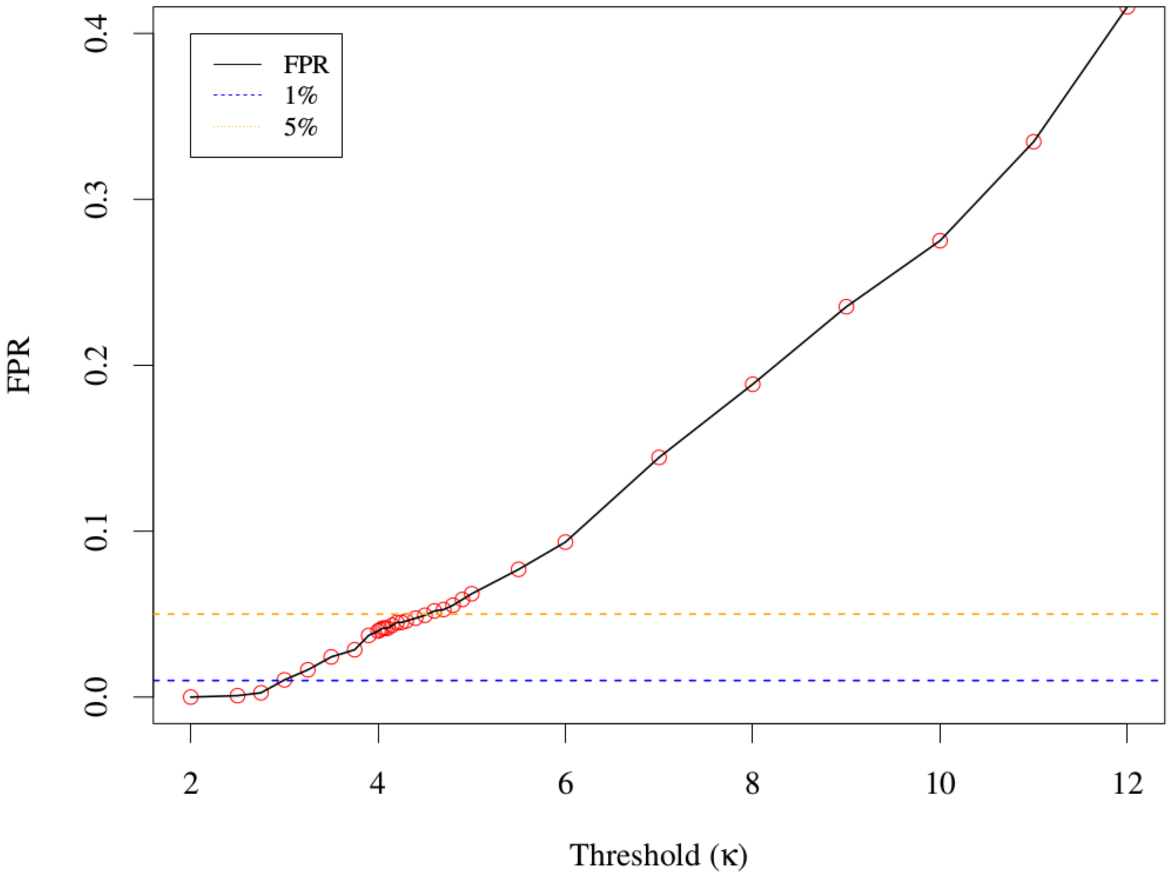
99.516% F-score

[1] S. Wendzel: Protocol-independent Detection of “Messaging Ordering” Network Covert Channels, in Proc. CUING, ACM, 2019.

Message Ordering Pattern: Detection (Ethernet) [1]

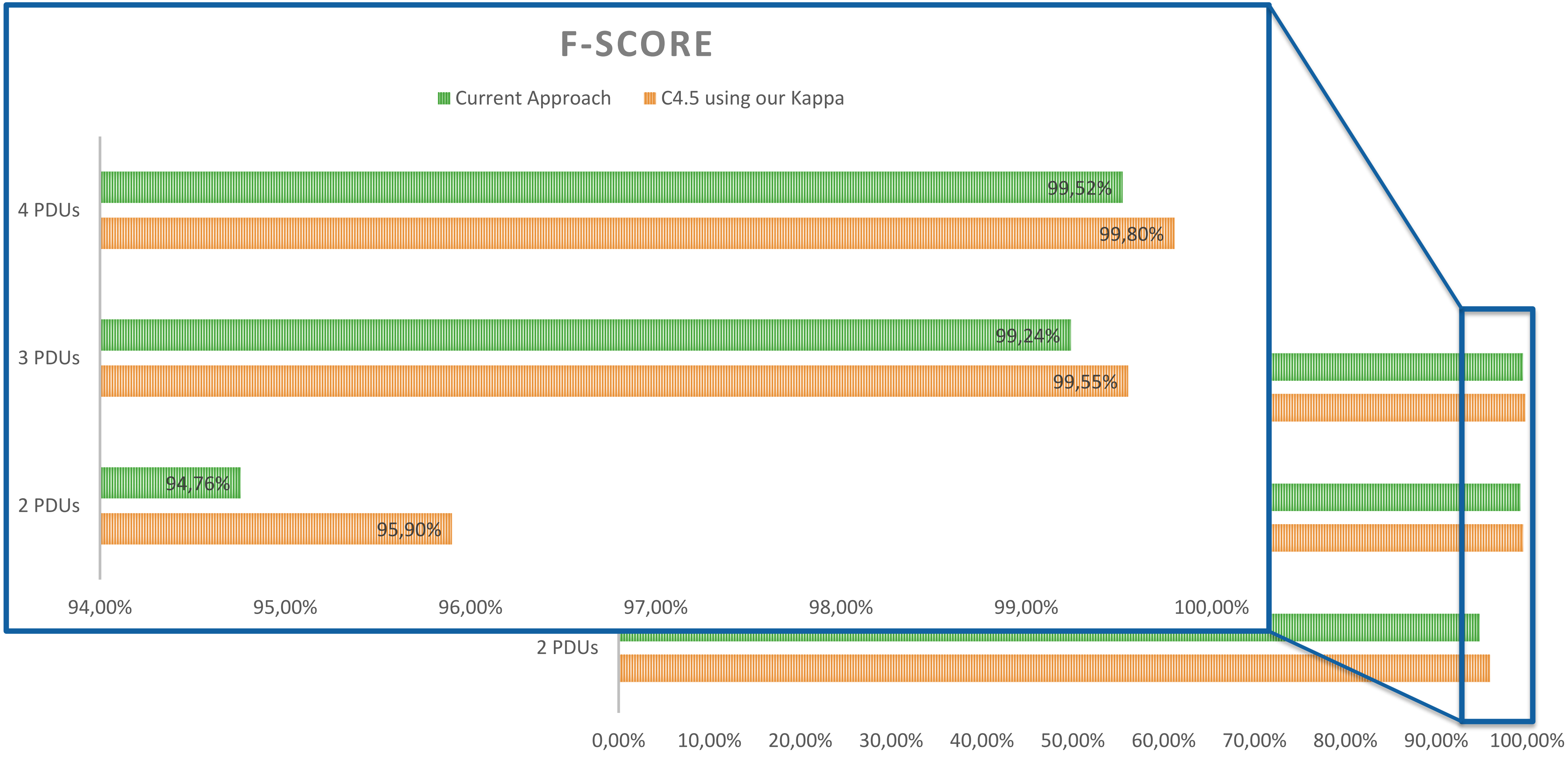
The optimal thresholds of κ = 2.75 to κ = 3.0 for channels using **3 or 4 PDUs** resulted in an FPR of **0.259%** to **1.038%**.

However, optimal threshold for **2 PDU** channels resulted in an FPR of **5.19%**.



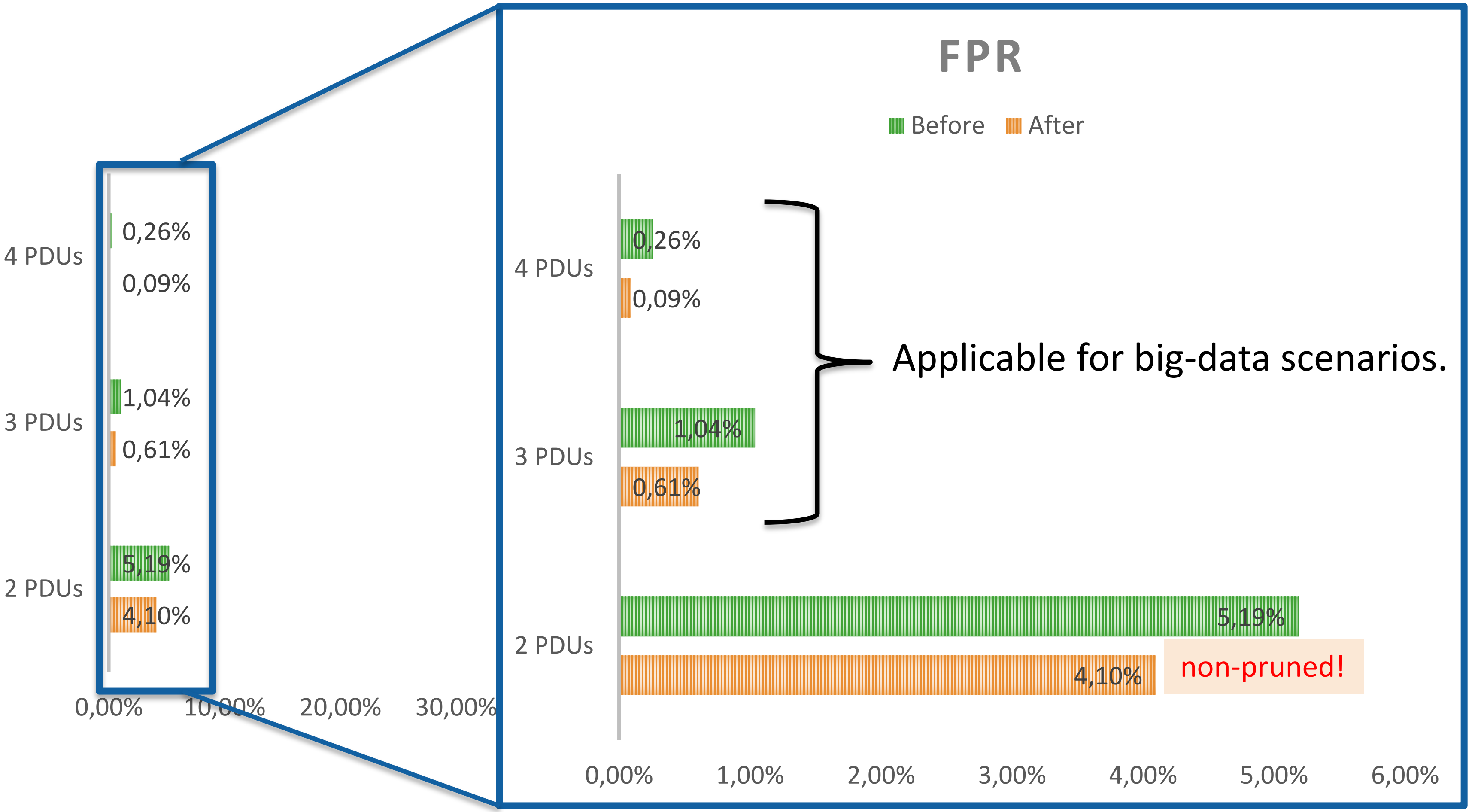
[1] S. Wendzel: Protocol-independent Detection of "Messaging Ordering" Network Covert Channels, in Proc. CUING, ACM, 2019.

Can we reduce the FPR further? [1]



[1] S. Wendzel: Protocol-independent Detection of “Messaging Ordering” Network Covert Channels, in Proc. CUING, ACM, 2019.

FPR with C4.5-determined Kappa thresholds [1]



Directly applicable to our heuristic:

4 PDUs: C4.5 selected K=2.5905 (instead of K=2.75). FPR -= 0.17%.

3 PDUs: K=2.8866 (instead of K=3). FPR -= 0.43%.

Not directly applicable to our heuristic:

2 PDUs: C4.5 found the same threshold, i.e. no improvement. However: non-pruned tree: FPR -= 1,09%.

[1] S. Wendzel: Protocol-independent Detection of “Messaging Ordering” Network Covert Channels, in Proc. CUING, ACM, 2019.

Another Countermeasure Variation

So far, we performed countermeasure variation for
- Compressibility Score
- ε-similarity
- *Regularity*

Each in combination with the following patterns:
- Size Modulation
- Artificial Re-transmission
- *Sequence Modulation*
- *Value Modulation*

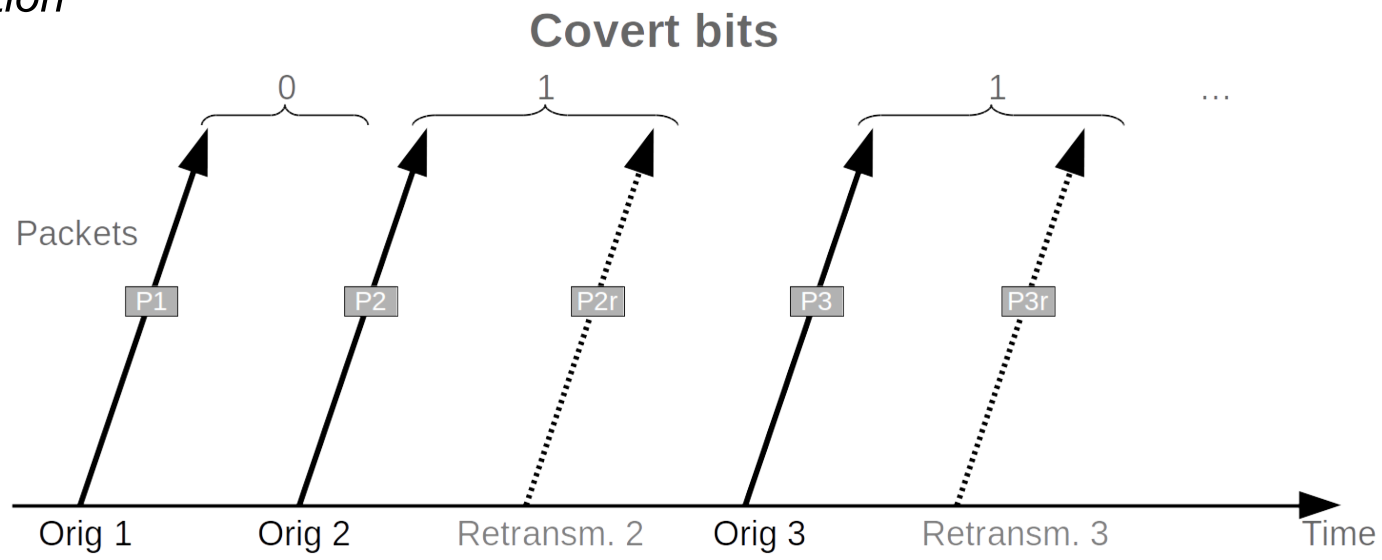
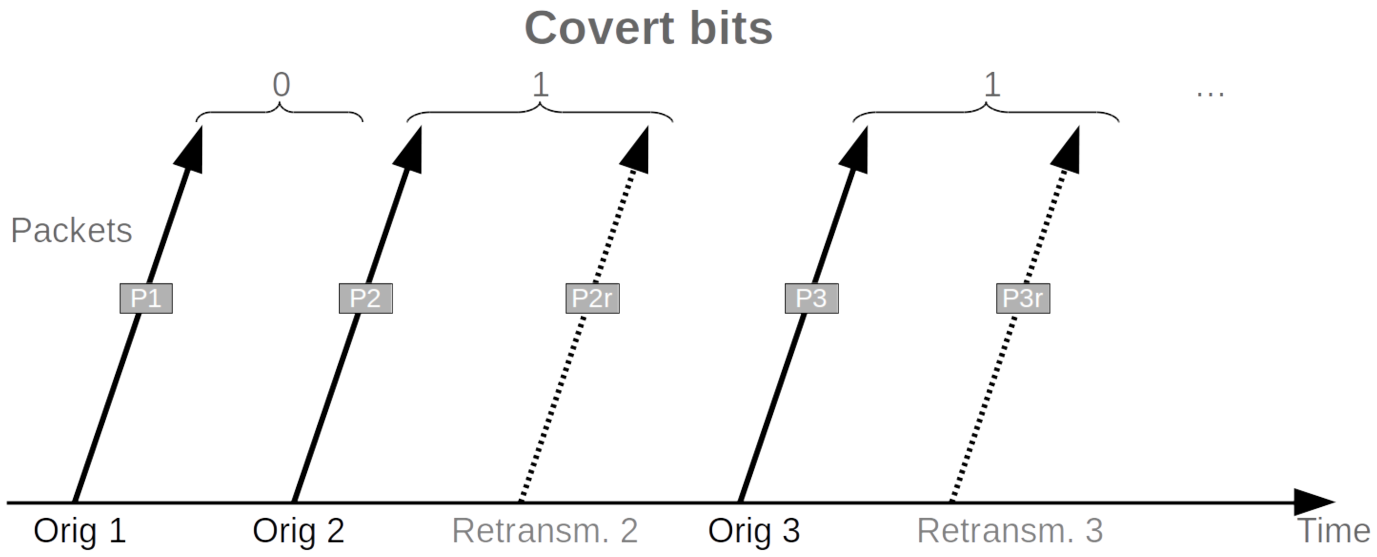


Fig.: S. Zillien, S. Wendzel: [Detection of Covert Channels in TCP Re-transmissions](), in Proc. NordSec'18, Springer, 2018.

Countermeasure Variation for the Artificial Re-transmission Pattern [1]

- Using TCP re-transmissions
- To match traffic patterns, we
  - studied typical re-transmissions of Internet traffic (different routes; repeated measurements several times for each route; at different days/hours), and
  - adjusted and optimized our CC to legitimate traffic's characteristics (very low transmission rate to increase covertness; robust coding).



[1] S. Zillien, S. Wendzel: Detection of Covert Channels in TCP Re-transmissions, in Proc. NordSec'18, Springer, 2018.

Countermeasure Variation for the Artificial Re-transmission Pattern [1]

ε-similarity

Input modifications:

Succeeding retransmission's sequence numbers

Modification of detection algorithm:

- Adjust thresholds for detection.

Compressibility

Input modifications:

Succeeding retransmission's sequence numbers

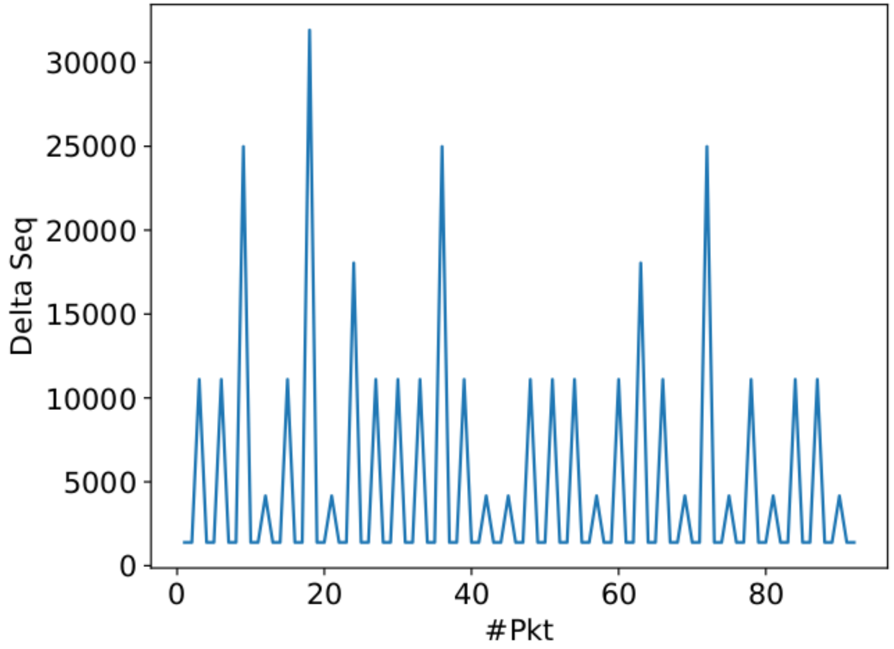
Modification of detection algorithm:

- Replace IAT-to-ASCII string conversion with new algorithm so that it can deal with 32-bit unsigned int.
- Adjust thresholds for detection.

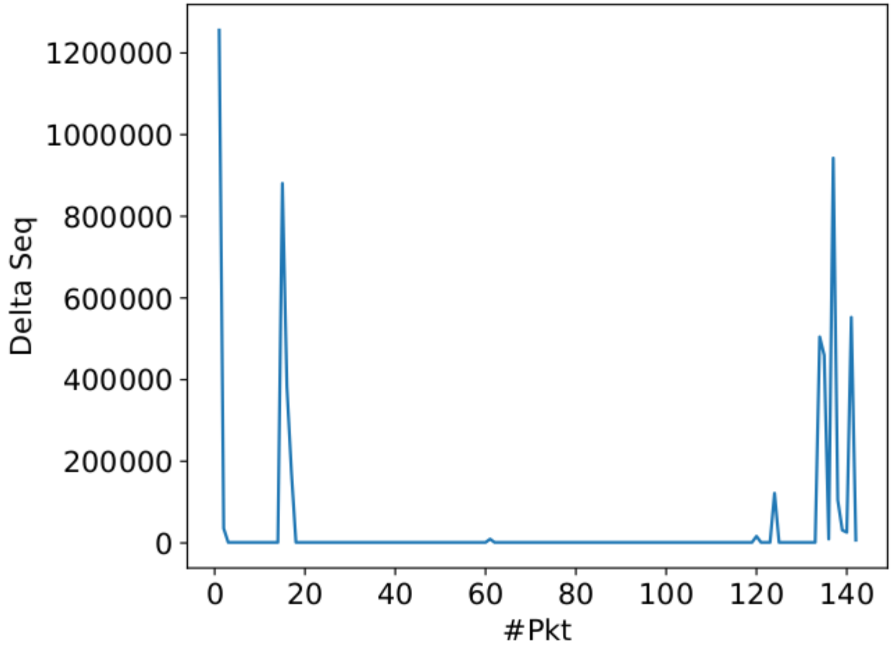
[1] S. Zillien, S. Wendzel: Detection of Covert Channels in TCP Re-transmissions, in Proc. NordSec'18, Springer, 2018.

Countermeasure Variation for the Artificial Re-transmission Pattern [1]

Results for ε-similarity (figures from [1]):



(a) Typical Covert channel traffic



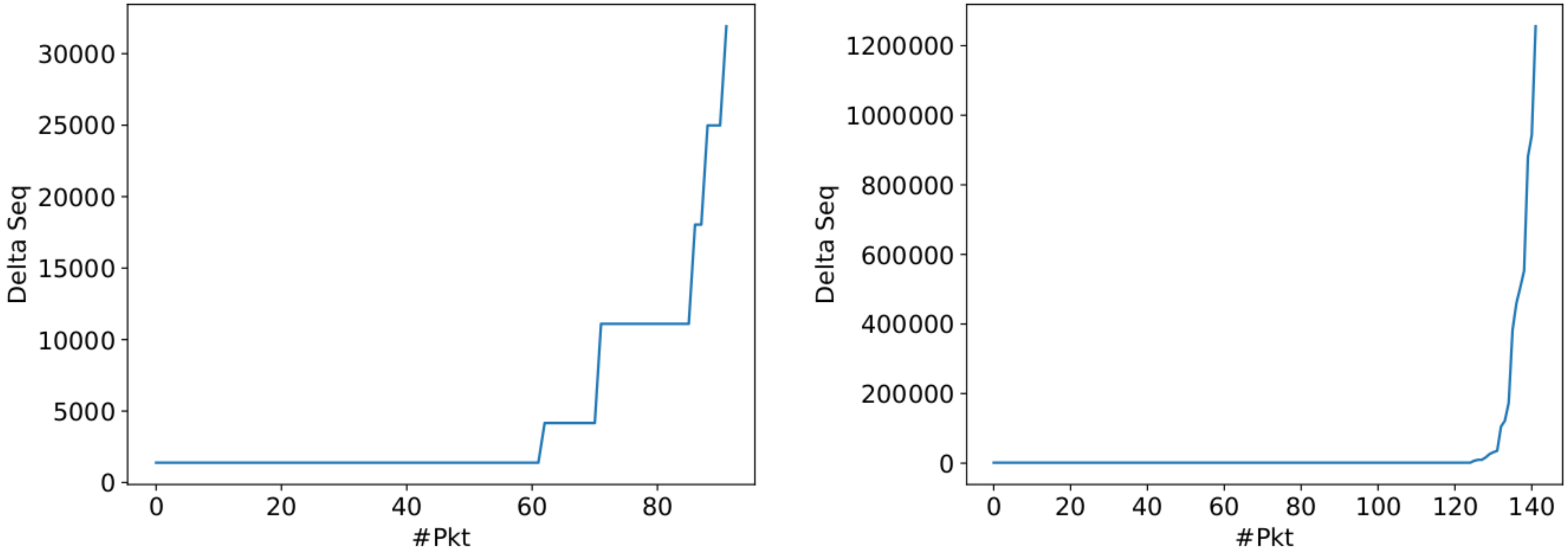
(b) Regular traffic (Germany 2)

Comparison: covert - regular: Δ values between retransmissions

[1] S. Zillien, S. Wendzel: Detection of Covert Channels in TCP Re-transmissions, in Proc. NordSec'18, Springer, 2018.

Countermeasure Variation for the Artificial Re-transmission Pattern [1]

Results for ε-similarity (figures from [1]):



(a) Typical Covert channel traffic

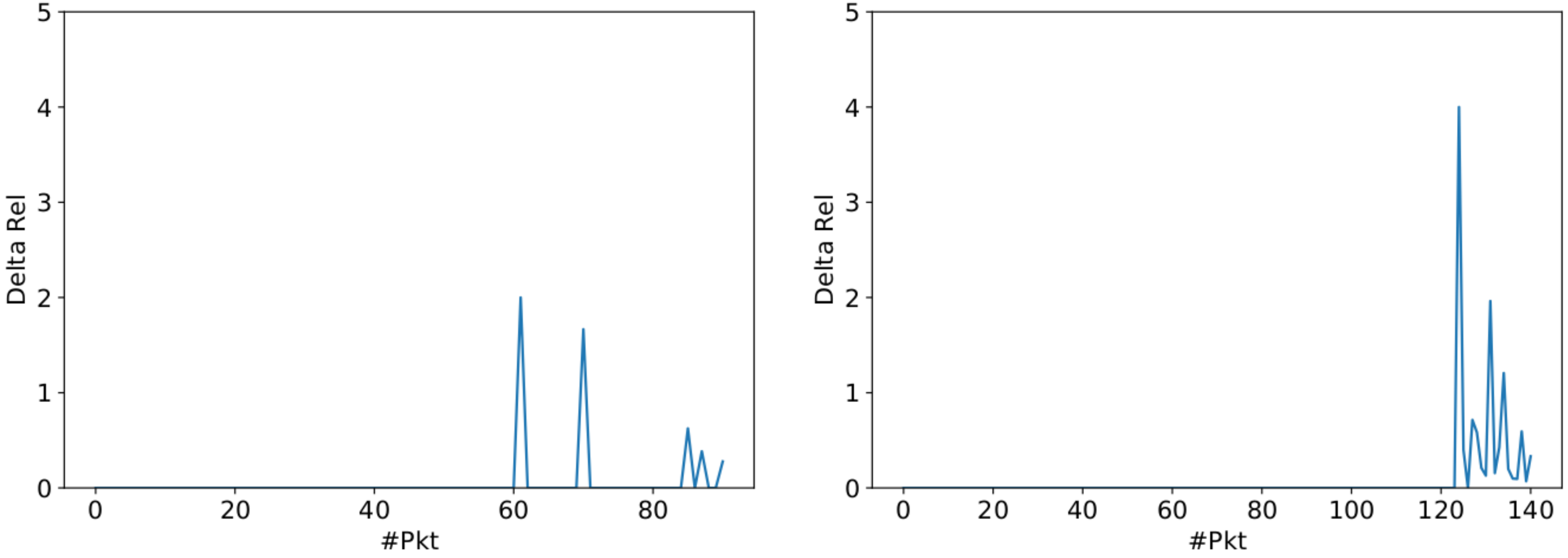
(b) Regular traffic (Germany 2)

Comparison covert - regular: sorted Δ values between retransmissions

[1] S. Zillien, S. Wendzel: Detection of Covert Channels in TCP Re-transmissions, in Proc. NordSec'18, Springer, 2018.

Countermeasure Variation for the Artificial Re-transmission Pattern [1]

Results for ε-similarity (figures from [1]):



(a) Typical Covert channel traffic

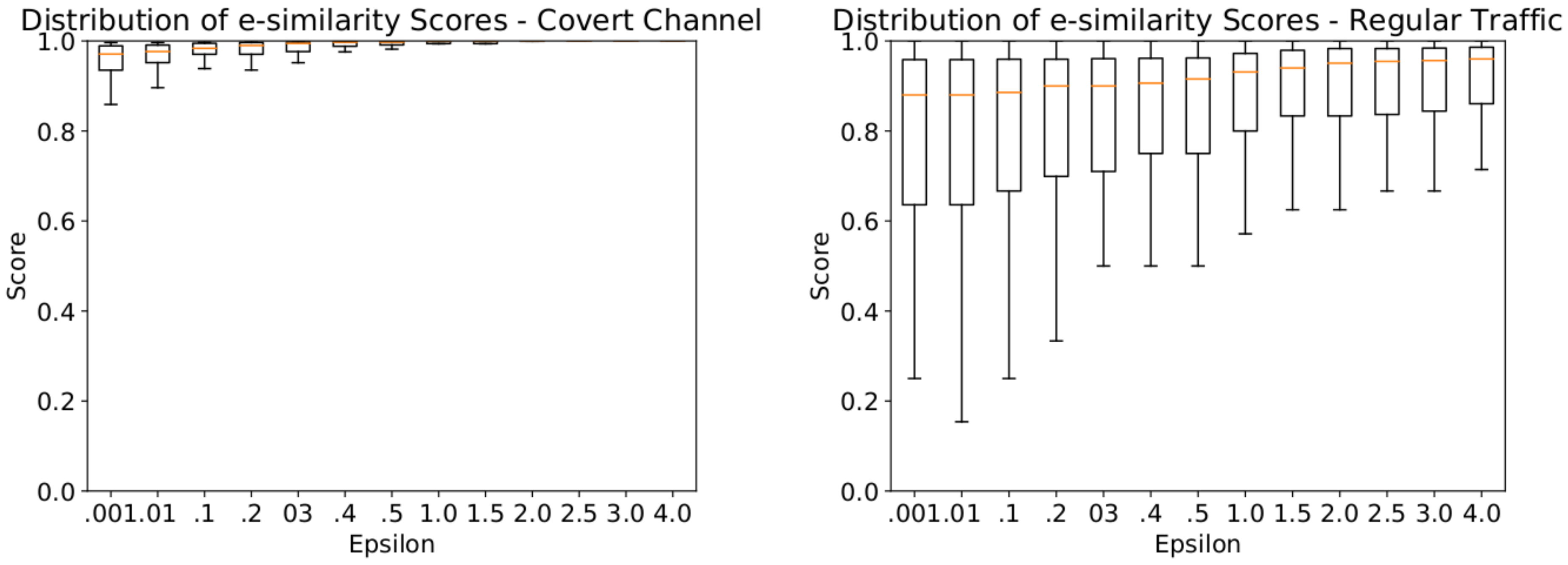
(b) Regular traffic (Germany 2)

Comparison covert - regular: relative differences of λ values

[1] S. Zillien, S. Wendzel: Detection of Covert Channels in TCP Re-transmissions, in Proc. NordSec'18, Springer, 2018.

Countermeasure Variation for the Artificial Re-transmission Pattern [1]

Results for ε-similarity (figures from [1]):



(a) Covert channel traffic

(b) Regular traffic

[1] S. Zillien, S. Wendzel: Detection of Covert Channels in TCP Re-transmissions, in Proc. NordSec'18, Springer, 2018.

Countermeasure Variation for the Artificial Re-transmission Pattern [1]

Results for ε-similarity (extracted from [1]):

Results (mixed covert channels vs. mixed regular traffic): We chose $\epsilon = 0.01$ with an upper threshold of 0.997 (no lower threshold), $\epsilon = 0.2$ with a lower threshold of 0.95 and $\epsilon = 2.5$ with a lower threshold of 1.0 (both no upper threshold).

Detection results - ϵ-similarity

		Actual Class	
		Covert Channel	Regular Traffic
Detected Class	Covert Channel	154	1
	Regular Traffic	6	130

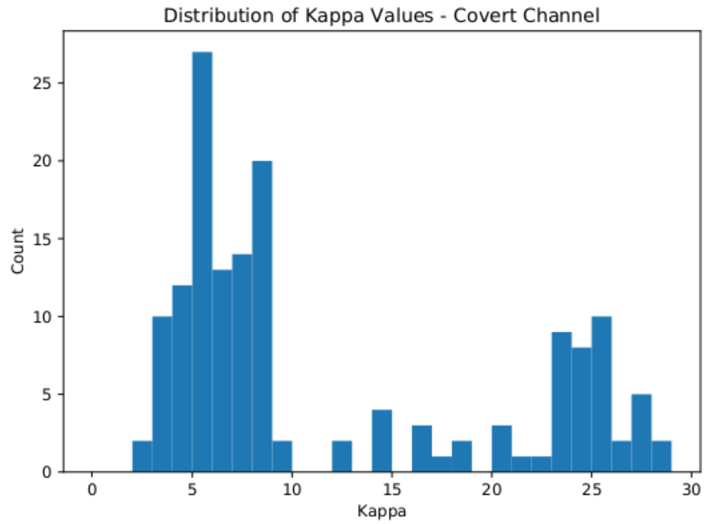
Please note that we focused solely on the detection of an optimized covert channel.
Also, the remaining undetectable channels were those configured using large gaps $D \geq 500$ between retransmissions combined with extremely few retransmissions (≤ 27) (resulting anyway in a short transmission and low transmission rate).

[1] S. Zillien, S. Wendzel: Detection of Covert Channels in TCP Re-transmissions, in Proc. NordSec'18, Springer, 2018.

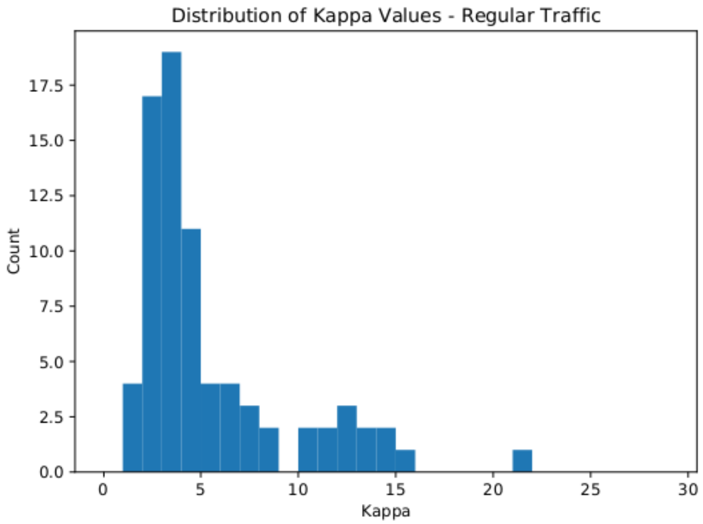
Countermeasure Variation for the Artificial Re-transmission Pattern [1]

Results for compressibility (figures from [1]):

Compressibility worked not so well (values of legitimate and covert traffic are quite overlapping; performs better with longer input data, i.e. more retransmissions)



(a) Covert channel traffic



(b) Regular traffic

However, channel was an optimized one. Better results for trivial retransmission channels.

[1] S. Zillien, S. Wendzel: Detection of Covert Channels in TCP Re-transmissions, in Proc. NordSec'18, Springer, 2018.

Countermeasure Variation for the Artificial Re-transmission Pattern [1]

Results for compressibility (extracted from [1]):

Using an exemplary threshold $\kappa = 6$, we obtained the following detection results:

Detection results - compressibility

		Actual Class	
		Covert Channel	Regular Traffic
Detected Class	Covert Channel	136	26
	Regular Traffic	24	51

[1] S. Zillien, S. Wendzel: Detection of Covert Channels in TCP Re-transmissions, in Proc. NordSec'18, Springer, 2018.

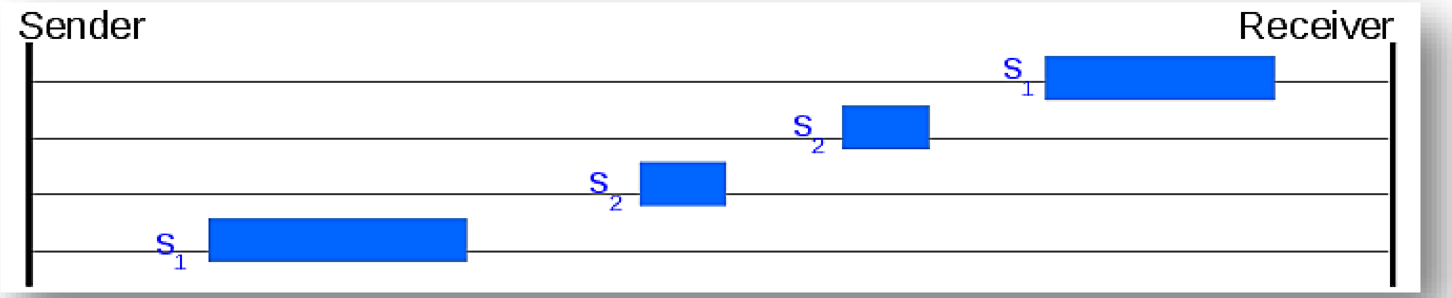
Size Modulation Pattern

So far, we performed countermeasure variation for

- Compressibility Score
- ε-similarity

Each in combination with the following patterns:

- Size Modulation
- Artificial Re-transmission
- *Sequence Modulation*
- PDU Order



Size Modulation Pattern: Compressibility Score [1]

- **Compressibility Score:**
  - Size Modulation pattern detectable
  - But: Compressibility scores highly depending on covert channels' configuration (figures from [1])

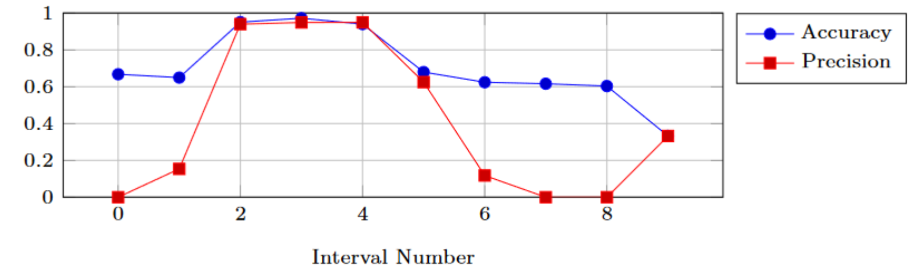


Figure 2: Precision and accuracy for covert channels using the payload sizes 1,000 and 1,001 bytes.

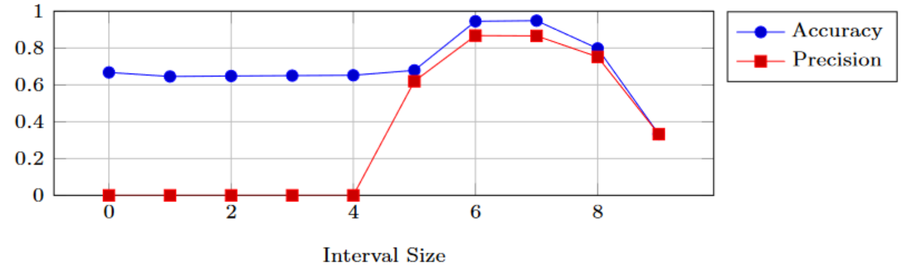


Figure 3: Precision and accuracy for covert channels using the payload sizes 50 and 60 bytes.

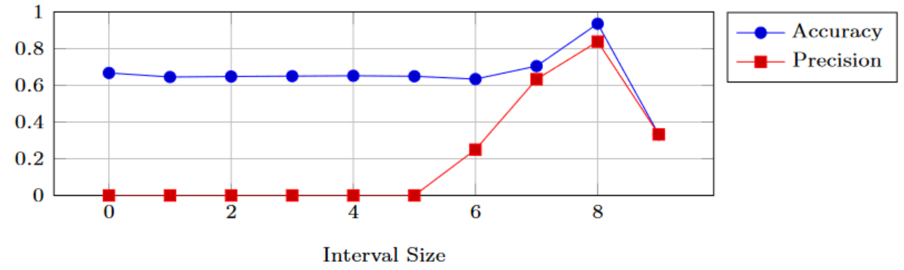


Figure 4: Precision and accuracy for covert channels using the payload sizes 100 and 200 bytes.

[1] S. Wendzel et al.: Detection of Size Modulation Covert Channels Using Countermeasure Variation, Journal of Universal Computer Science (J.UCS), Vol. 25(11), pp. 1396-1416, 2019.

Size Modulation Pat
Compressibility Scor

- **Compressibility Score:**
  - What happens overall, or if more than two symbols are transferred? (figures from [1])

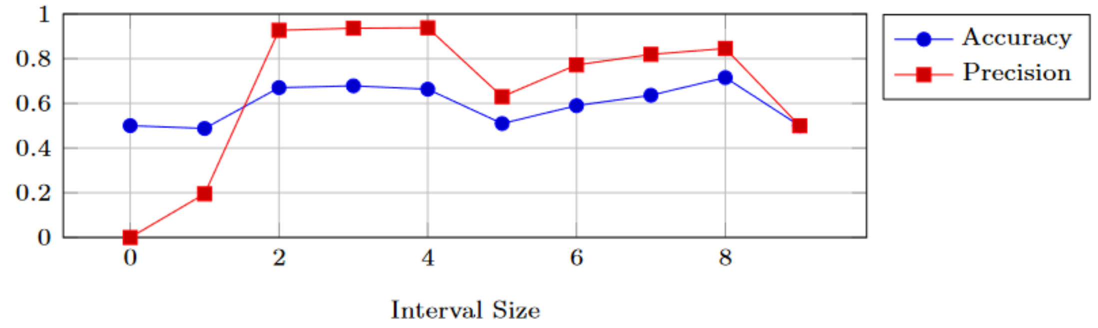


Figure 5: Precision and accuracy for a mixture of *all* two-symbol covert channels.

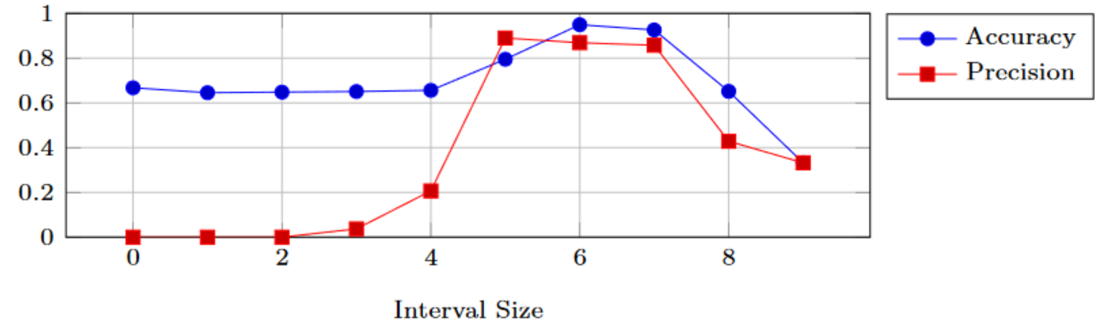


Figure 6: Precision and accuracy for covert channels using the payload sizes 100, 200 and 300 bytes.

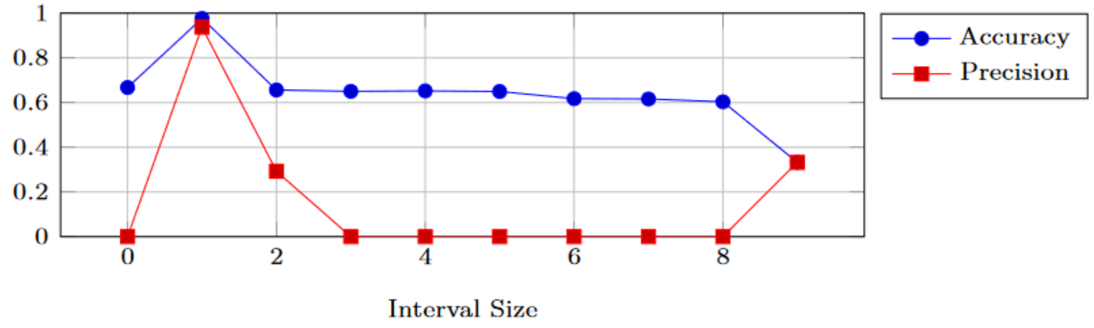
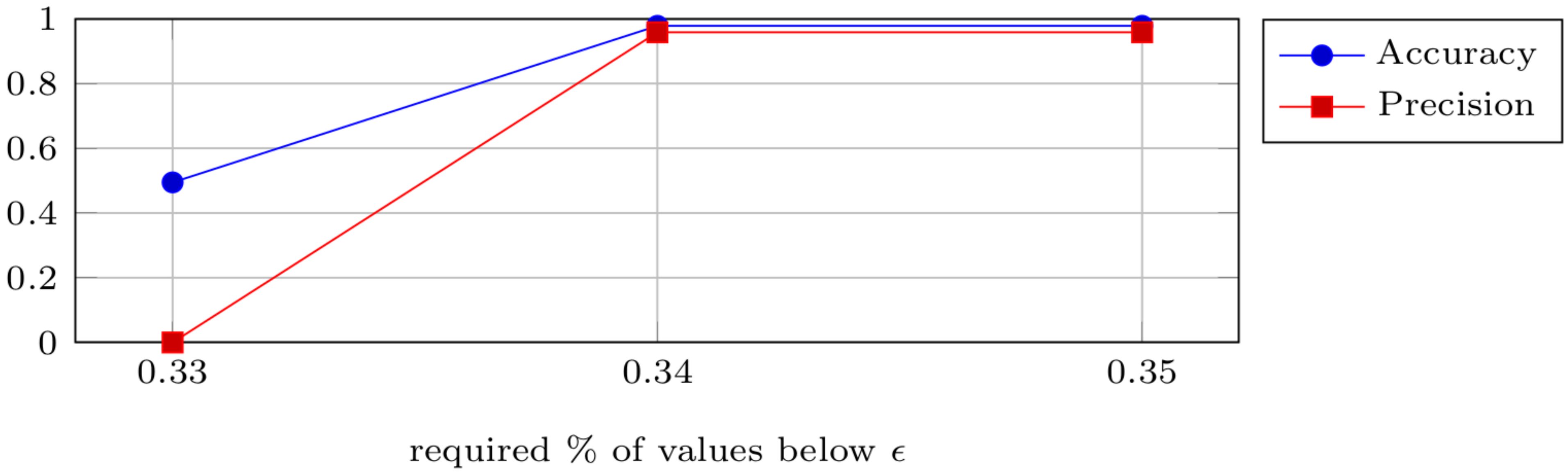


Figure 8: Precision and accuracy for covert channels using the payload sizes 100 to 800 bytes (in steps of 100 bytes).

[1] S. Wendzel et al.: Detection of Size Modulation Covert Channels Using Countermeasure Variation, Journal of Universal Computer Science (J.UCS), Vol. 25(11), pp. 1396-1416, 2019.

Size Modulation Pattern: ε-Similarity [!]

- **ε-Similarity:**
  - Able to detect two-symbol covert channels with an accuracy of 97.8% and precision over 95.8% (FPR: 4.36%).
  - Not able to detect covert channels with 3+ symbols.
  - Figure: 2-symbol covert channel (1000 and 1001 bytes) from [1]:



[1] S. Wendzel et al.: Detection of Size Modulation Covert Channels Using Countermeasure Variation, Journal of Universal Computer Science (J.UCS), Vol. 25(11), pp. 1396-1416, 2019.