Journal of Universal Computer Science, vol. 30, no. 8 (2024), 1048-1067
submitted: 29/6/2023, accepted: 9/2/2024, appeared: 28/8/2024 CC BY-ND 4.0

Archive-based Covert Channel in Sensor Streaming Data

Carina Heßeling
(FernUniversität in Hagen, Hagen, Germany
https://orcid.org/0000-0002-9364-1282, carina.hesseling@fernuni-hagen.de)

Sebastian Litzinger
(FernUniversität in Hagen, Hagen, Germany
https://orcid.org/0000-0003-2200-7337, sebastian.litzinger@fernuni-hagen.de)

Jörg Keller
(FernUniversität in Hagen, Hagen, Germany
https://orcid.org/0000-0003-0303-6140, joerg.keller@fernuni-hagen.de)

Abstract: With the advent of the Internet-of-Things, sensor data streams are widespread and thus a promising target for network steganography including covert channels. Therefore, we present an approach where the covert sender and receiver first build an archive of values that occur in the stream over a time interval, and then encode bits of the secret message via sensor stream values belonging to the class of seen values or not. If a sensor value's category does not match the required bit value, it is adapted to switch its category. The chosen value minimizes the distance to the original sensor value to improve stealthiness. Furthermore, we present an application scenario and an analysis of the channel's detectability, in particular how detectability and steganographic bandwidth can be traded against each other. We have tested the approach and the proposed detection metrics via simulation experiments.

Keywords: Covert channel, network steganography, sensor data transmission, countermeasures
Categories: C.2.0, B.2.0, K.4.2, D.4.7
DOI: 10.3897/jucs.108811

1 Introduction

With the rise of the Internet-of-Things that ranges from climate monitoring via industry plants to smart cities [Santos et al., 2018], sensors are ubiquitous and provide streams of sensor data which might be numerical or symbolic. Edge and ad-hoc networks transport these data streams from sensors to base stations for further processing and aggregation. As sensors are computationally (and physically) constrained devices, a sensor might be compromised and thus leak data that are normally not transmitted and processed. As transmission frequently occurs via wireless channels, and often multi-hop via other sensors, a compromised sensor might also spy on data from other sensors and leak those out.

The wireless bandwidth is small and sensors have limited energy supply. Thus, extra transmissions to send leaked data may not be possible, or at least be very suspicious. As a consequence, the leaking of data must happen as part of the normal data transmission. It can e.g. occur by compressing the normal data in some way and use the freed bits to transmit the secret information, thus forming a steganographic or covert transmission

channel. In the present work, we will investigate a covert channel with a different approach. In a first phase, covert sender and receiver monitor data streams and build an archive of seen values. In a second phase, a transmitted value that is in the archive, i.e., which has previously been seen, represents a 1, otherwise it represents a 0. If a sensor value does not correspond to the next bit of the secret message, the sensor value is modified by choosing a value from the complementary set. The choice is such that the distance between the original and modified values is minimal, to avoid suspicion. As any distance metric can be chosen, e.g. Levenshtein or Needleman-Wunsch [Ramsden, 2015], the proposal is not restricted to numerical values but can also be used for transmitted character or symbol strings. While on average the number of modifications will not be smaller than a simple replacement of the least significant bit (we restrict now to numeric data) by the next bit of the secret message, the technique allows to adapt and thus should be less detectable with statistical means than the simple way.

Next to proposing a new covert channel technique, we also investigate limitations, countermeasures and detectability of this proposal, as well as present an application scenario. The detectability is checked also via simulation experiments.

To summarize, we provide the following contributions[1].

- We propose a storage covert channel in sensor data transmission, which adapts to the data values that are transmitted and signals secret data transmission via previously unseen data values, a combination of techniques that to the best of our knowledge has not been proposed in the literature so far.
- We provide a prototype implementation that we apply on sensor data streams taken from a real-world application; with these we perform simulation experiments on detectability via entropy and compressibility metrics and demonstrate the common trade-off between steganographic bandwidth and detectability, i.e., demonstrate that the covert channel can adapt to more suspicious environments, or put otherwise, that detection means can help at least to limit bandwidth of such covert channels.
- We describe an example application scenario to concretize where covert sender and receiver might be placed and which knowledge a warden would need to have to perform the detection experiments that we used in our simulations; we also discuss countermeasures that go beyond pure detection-based approaches.

The remainder of this article is structured as follows. Section 2 summarizes background information on network steganography, sensor data streams and related work. In Section 3, we present and analyze our archive-based proposal for a storage covert channel in sensor data. Section 4 sketches an application scenario and scenario-dependent possible countermeasures. Section 5 investigates the detectability of our proposal when applied to real-world sensor data in simulation experiments. Section 6 presents conclusions and an outlook to future work.

2 Background

2.1 Network Steganography

Steganography investigates possibilities to hide the existence of some secret information, in contrast to cryptography where the content of that information is protected from

[1] A preliminary version of this research is part of the PhD thesis of one of the authors [Heßeling, 2023].

unwanted reading [Mazurczyk et al., 2016]. The secret information may be stored or may be transmitted. We focus on the case where the secret information is digital and is hidden within other digital information. In network steganography, two communication partners, called overt sender and receiver (OS and OR, respectively), communicate via a communication network. Their communication is called overt communication or carrier. Two further entities, the covert sender (CS) and the covert receiver (CR) are present in the network. They may either reside with overt sender and receiver, or may be placed on the communication path, cf. Figure 1.

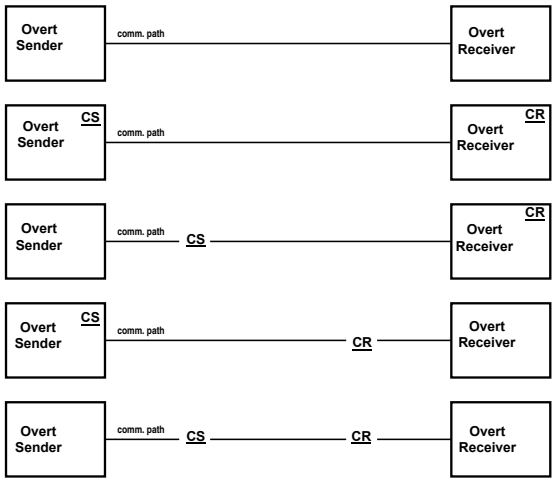

Figure 1: Different possibilities to position covert sender and receiver on the communication path between overt sender and receiver.

CS wants to send a secret message to CR, i.e., without any warden noticing that this message has been sent. The warden may sit at any place on the communication path, or may reside with the overt receiver. Typically, the secret message is additionally encrypted, so that it looks like a random bitstring, and remains confidential in the case of detection.

CS hides the secret message in an innocent communication message originating from the overt sender (we ignore the case that CS might trigger the sending of such message). For example, an unused header bit in a network packet can be used to transfer a bit of the secret message, or the payload of packets might be modified to transmit the secret message. CS and CR thus establish a storage covert channel. There are also timing and hybrid channels [Mazurczyk et al., 2016], however we will not use these types of channels in the present work. A covert channel is a policy-breaking communication that was unforeseen in the original communication system [Lampson, 1973].

CS and CR might assume different positions in the communication network. CS could be placed at the overt sender, i.e., it has compromised that machine, and can at least partly influence the decisions that the overt sender takes when participating in the communication protocol. Alternatively, CS can be placed on the communication path

between overt sender and receiver, i.e., it can only intercept and modify packets sent by the overt sender, but has no influence or knowledge of sender internals that led to certain values or decisions. CR can be placed at the overt receiver or on the communication path in-between. In both cases, CR might try to restore the original content of the overt communication, to reduce detectability. Otherwise, it will simply intercept the overt communication and decode the secret message hidden in it.

Network covert channels can be classified with patterns [Wendzel et al., 2021] and also according to the properties steganographic bandwidth, robustness against modifications on the communication path, and detectability [Mazurczyk et al., 2016].

2.2 Sensor Data Transmission

Networks of sensors are used in a multitude of application areas ranging from smart cities to climate monitoring [Santos et al., 2018]. Before the advent of the Internet-of-Things (IoT), sensors specific for a given purpose were often distributed over an area, i.e. placed in unprotected places. Thus, the sensors need to communicate wirelessly, hence the term wireless sensor network. With IoT, sensors could also be mobile or stationary devices of any kind. The typical communication mode is to forward sensor data to a sink, which is a stationary computer with Internet connection. Sensors, especially if battery-powered, are normally weak devices, and hence only have very limited resources to protect data that is stored or transmitted. They might be tampered with (cf., e.g., [Becher et al., 2006]) or otherwise compromised, e.g., by a malicious code update [Wang et al., 2006], for example, to leak some data or acquire additional data. In both cases, the secret data must be brought to the sink, i.e. transmitted as well. However, separate transmissions would deplete the sensors' energy resources, or at least be suspicious. Thus, it might be preferable to transmit such data as part of normal transmissions.

2.3 Related Work

Modification of network packets' payloads to transport secret message data has been investigated e.g. in [Jankowski et al., 2013, Mazurczyk et al., 2014]. These works used compression to exploit redundancy inherently present in the payload data, and added secret message bits in the space freed by compression. In contrast, we modify numeric or textual payloads, in particular short payloads of few bytes. The modification is not to insert a bit of the secret message, but depends on whether values have been seen previously or not, where each class represents one bit.

Modification of numeric payloads has been investigated in [Heßeling and Keller, 2022], but they also shorten mantissa length in floats to transport message bits.

Also [Mazurczyk et al., 2018] investigate payload modifications, and mention to modify the least significant bit (LSB) of a numeric value (see their Fig. 6). In contrast, we use a general definition of distance between between original and modified value and use two complementary sets of values, where the modified value will belong to a different set than the original value, which allows to adapt to specific situations (see next section).

The LSB approach has also been used in image steganography of raster images, yet has also been overcome by more advanced approaches that e.g. adapt which pixels to use [Fridrich, 2009]. In contrast to our approach, the complete image is known beforehand.

Finally, the LSB approach has been used in network steganography, however with modification of header fields of network packets, such as TTL (time to live) [Mazurczyk et al., 2016].

[Tuptuk and Hailes, 2015] describe a covert channel where a compromised sensor node modifies the value of sensor data to be transmitted so that it is close to the original data but the LSB of the encrypted modified value is set according to the secret message bit to be transmitted. In contrast, we modify data values so that they are close to each other and the modified value belongs to the desired class "previously seen" or "not yet seen" as demanded by the secret message bit. [Patuck and Hernandez-Castro, 2013] investigate hidden channels in the Extensible Messaging and Presence Protocol (XMPP). While the name of the protocol sounds similar to MQTT, XMPP refers to text messages, whereas we have focused on numerical data. The proposed channels either modify header data, and those that work on payload perform text-specific modifications like adding blank spaces. A different approach is taken in [Ho, 2019]. There, a statistical test on the transmitted data is used, and the bit value of the secret message is determined depending on whether the null hypothesis for that test is accepted on the data or not. Again, this approach does not explicitly take into account the data values that have already been seen in the past.

Covert channels to leak data from sensors have also been proposed on the physical level. A physical covert channel where devices without Internet connection leak information is presented by [Li et al., 2022]. It works via influencing device fans that are sensed by millimeter waves. In contrast, our approach uses Internet connection and modifies application-layer values prior to transmission. A covert channel on the physical layer of communication is proposed by [Hou et al., 2023]. While they modify the modulation of the LoRa scheme, our approach works on the application layer, by modifying payload values prior to transmission.

The use of prior knowledge to improve encoding of information has been used previously. Dynamic approaches are known both in frequency-based coding such as Huffman coding, see [Vitter, 1987], and in arithmetic coding, cf. [Marpe et al., 2003]. For Huffman codes, even memory-efficient storage for very large symbol sets have been investigated in [Pigeon and Bengio, 1998]. However, those approaches are adaptive over the whole time of operation, while in our approach, there is a first phase where we monitor symbols that appear, and in a later phase only distinguish between symbols that have been seen in the first phase and symbols that have not appeared in the first phase, without further adaptation and without reference to their frequencies. Beyond that, the covert receiver does not see the same set of symbols as the covert sender in the second phase, as some symbols have been modified by the covert sender.

3 Covert Channel Proposal

3.1 Setting

We suppose that we have a stream of sensor data values in a wireless network, for which we propose a covert channel that builds on data values of the recent past. As an example, the stream of data values may be transmitted via an MQTT channel [Banks et al., 2019] as in [Heßeling et al., 2022]. In our proposal, CR does not have to be able to undo modifications done by CS, but just intercepts the transmitted data values. Thus, CR can be placed on any part of the communication path, or with the overt receiver. We assume that CS resides with the overt sender.

3.2 Archive-based Covert Channel Approach

The approach to be described is non-reversible. Thus, CR need not undo any changes made to the data stream by CS. CR just listens to the transmitted sensor values and interprets every value as agreed on in advance with CS. As preparation, CS and CR both monitor and analyze all transmitted numbers for an appointed period of time λ. Assuming that there are various different sensors sending their measurement data, CS and CR agree on how many and which sensors to use for their purpose. This decision might be made even after the monitoring phase via an out-of-band channel as a form of optimization, i.e., to use the most suitable sensor stream or streams. Thus, this preparation phase could also be interpreted as a kind of network environment learning phase as introduced in [Yarochkin et al., 2008], although the initial approach monitored if certain covert channel possibilities, e.g., certain ports, were blocked or not, while the approach presented here monitors payload data values.

After finishing these preparations, CS begins with sending the secret message. There are two cases to be considered, depending on the next bit of the secret message to be transferred (cf. Figure 2): If it is equal to 1, CS checks if the next value to be transferred by OS is in the collection of values observed during time period λ. If so, CS changes this value to a value not yet seen. If the actual value is not in its collection of already transferred values, it leaves it unchanged. If the next bit of the secret message is equal to 0, CS also checks if the value is a new one, or has already been transferred during λ by checking if its prepared collection contains this current value. If so, the number remains as it is to be transferred. If it is a new value, CS changes it to a value of its collection.

In both cases, the value within the complementary set is chosen such that the distance between modified and original value is minimized. The definition of distance is dependent on the data type of the transmitted values. If they are numeric, then the absolute value of their difference can be used. If they are bitstrings, the Hamming distance might be used, or a more complex distance computation like Needleman-Wunsch or Levenshtein can be used for symbol sequences.

Moreover, the distance might also include additional constraints. If values in a sequence have some correlation (e.g. in 80% of cases an even value is followed by another even value), then the distance might include this, too. If the complementary set is large, then a minimization might be too computationally intensive, and the requirement of minimum distance might be replaced by sufficiently small distance.

Modifying a sensor value requires computations by CS as it has to determine a value with minimal distance to the given datum and the property of either being in the archive or not. The archive, i.e., the set of values already seen, is represented explicitly, while the complementary set is not, normally being much larger. Hence, the former case is not computationally expensive: assuming the archive is sorted, a simple bisection yields the required value. In the latter case, one needs to check values in order of ascending distance to the given datum for membership in the archive until the check comes back false. This can be hard to accomplish in a timely and inconspicuous manner at runtime. Fortunately, once the archive has been completed, one can statically compute a mapping of archive values to (sets of) closest values not in the archive, which can subsequently be applied when transmitting the secret message. What is more, computing such a mapping requires less effort per element than performing the operations described above for an individual datum: again assuming the archive is sorted, one processes the archive values one by one, only maintaining two consecutive values not in the archive, which are updated whenever the next value in the archive does not lie in between these two.

CR checks the incoming stream of numbers, cf. Figure 3: if it sees a number that

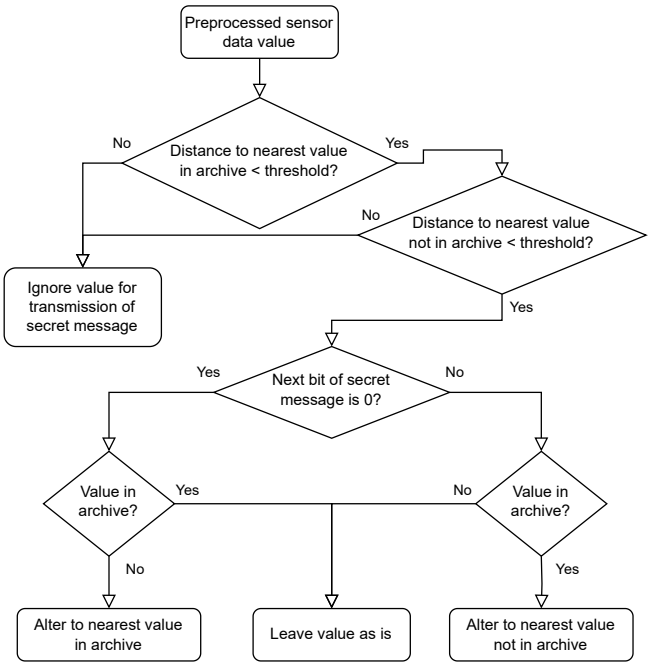

Figure 2: Covert sender activity for archive-based covert channel

is in its collection of values sent during λ, CR interprets it as 0 for the next bit of the secret message. If it is a new value, CR interprets this as a 1 for the next bit of the secret message.

Thus, each sensor value sent carries 1 bit of the secret message. CR does not need to undo any changes to the transmitted values, it only needs to intercept and interpret the sensor data. Naturally, the changes CS inflicted on the data will lead to errors. To bound the magnitude of this error, CS may refrain from utilizing a given datum for the transmission of a secret message bit in case altering the datum would require a considerable deviation from the original value. To this end, CS and CR agree on a maximum deviation threshold θ, and prior to processing a datum check whether a possible modification would exceed θ, in which case no secret message bit is transmitted. Ultimately, this procedure may decrease the error at the expense of covert channel bandwidth. In Section 5.2, we will examine how various combinations of λ and θ values influence error, covert channel bandwidth, and detectability.

CS and CR may also agree only to use a fraction, e.g., each t-th transmitted value, to transmit a bit of the secret message, in order to reduce introduced error and thus reduce the chance of detection.

CS and CR might even employ an error-correction block code on the secret message, and thus CS would be able from time to time to transmit the wrong bit value of the secret message in order to reduce the number of modifications and thus reduce detectability further. As error-correction codes add bits to the secret message, this would also reduce steganographic bandwidth further.

The presented approach may appear as a complicated form of a simple LSB covert

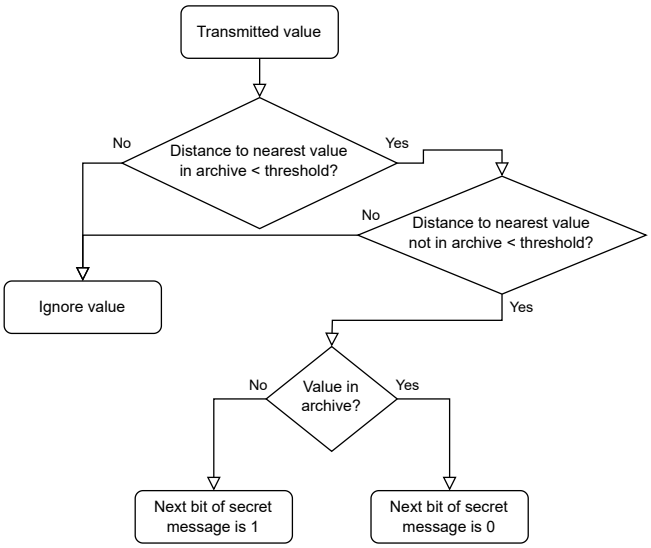

Figure 3: Covert receiver activity for archive-based covert channel

channel, where a bit of the secret message replaces the least significant bit of the sensor value. However, in this case the frequency of introduced errors is already determined by the distributions of zeroes and ones in the least significant bit of the sensor values and the bits of the secret message, with only minimal possibilities for changes (encrypt secret message or not, invert it or not). If the sensor value LSBs are independent of the secret message, then in 50% of the cases a bit will be flipped. In contrast, the choice of the set of archived values in our approach, together with the selection of a modified value from the complementary set, allows to reduce the frequency of modifications on the least significant bit. For example, if CS notes in the monitoring phase that a zero LSB in one sensor value is followed by another zero LSB in the next sensor value in 75% of the cases, then the corresponding value can be chosen accordingly to represent this. This comes at the price of possibly larger distances between original value and normal value. However, as long as the sensor values are not very far apart, they will not be seen as an anomaly. Moreover, our approach allows to also consider non-numeric data formats as carrier.

3.3 Analysis

As long as not all possible values have been seen in the first phase, the covert channel can operate in the second phase. Its detectability will depend on the distribution of values that have been seen in the first phase. Let us assume that all values seen in the first phase are from an interval $I = (a, b)$. If all possible values within this interval have appeared in the first phase, then in the second phase the modification of a value $x \in I$ will lead to result either a or b (depending on which half of I x comes from), making for a maximum distance of $(b - a)/2$ and an average distance of $(b - a)/4$ if the values are equi-distributed in I. If the values are normally distributed around $(b - a)/2$, the middle of the interval I, with shrinking variance the average distance goes towards $(b - a)/2$.

If a value outside I occurs, e.g., $b' > b$, and must be modified to a value within I, then b will be chosen in our example and the distance will be $b' - b$, i.e., it solely depends on the "outlier" b'.

If not all possible values in interval I have been seen in the first phase, e.g., because the sensor's resolution is lower than the resolution of the number format, then the maximum distance will be rather small during modifications in both directions.

4 Application Scenarios and Countermeasures

The application scenario that we have in mind is a network of sensors that uses, e.g., a lightweight publish-subscriber message transport protocol such as MQTT [Banks et al., 2019] for distribution of sensor values, although the covert channel proposed above can be used for any form of data transport of a data stream. In case of a compromised sensor node, CS modifies the sensor value, i.e., injects a bit of the secret message depending on the covert channel used, while the payload for the MQTT packet is packed. If CS sits on the communication path, the MQTT message could be intercepted and the payload modified. This can be possible as sensor nodes may be too weak to use AES encryption of packet payloads. Floating-point values, which we use in our experiments, are unknown to MQTT, thus the payload in this case will be binary data so that modification must only be accompanied by re-computation of the checksum to maintain integrity. If CR sits on the communication path, or has compromised the MQTT server (which is quite unlikely), then it can intercept the MQTT packets and decode the secret message. CR could also act as a subscriber to receive the MQTT messages, and decode the secret message, cf. Figure 4.

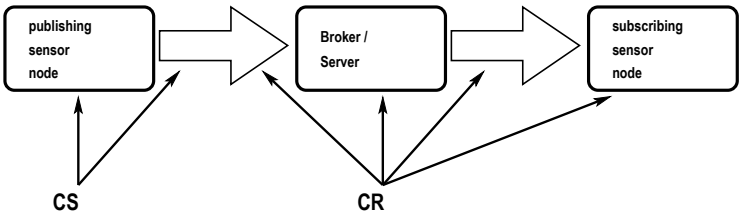

Figure 4: Possible positions of CS and CR on the communication path between publishing sensor nodes and subscribers of the published messages.

CS might symmetrically encrypt the secret message prior to embedding in the covert channel. This will lead to an even distribution of zeroes and ones in the bitstring that comprises the secret message. Furthermore, it provides some protection in case of detection by applying Kerckhoff's principle: even if a warden unveils the complete approach, the confidentiality of the secret message rests with the symmetric key [Menezes et al., 1996]. Yet, as the encryption might not be very strong to avoid computationally intensive operations by CS, the keyspace might be small (e.g. 56 bit with DES) so that a brute-force decryption by a warden might be feasible. If confidentiality is not needed, also other forms of randomization [Rajba et al., 2023] may be suitable.

A warden on the communication path between covert sender and receiver will be able to monitor anomalies which might occur as the distance measure will not be perfect.

However, as the data are sent as binary data in MQTT, the warden would not only need knowledge about MQTT but also about the data format present in the binary data payload of a certain MQTT channel. Thus, to perform this check, the warden must be not only application level but must even be aware of details of the particular instance of MQTT it is monitoring.

The presented covert channel does not violate any data format, but it might change data values. Hence, a warden cannot do more than the MQTT server or overt receiver, but would need the above-mentioned, special knowledge of the MQTT channel. Thus, we will concentrate on MQTT server and overt receiver in the following. They may be able to detect the covert channel as the error it introduces may destroy some correlation between succeeding values, or introduce artificial correlation. From our initial experiments (see next section), different heuristics might be useful in detecting this anomaly. However, the experiments also reveal that depending on the numeric characteristics of the sequence of data values, different heuristics may be necessary to spot the anomaly, so that a warden would need to employ several heuristics simultaneously. Yet, if the overt receiver is located far away from the sensor, e.g., in the cloud to have enough computing power, then some aggregation of data streams might have occurred in an intermediate node [Zhang, 2022], so that the above characteristics are further weakened and detection is less likely.

If CS is on the communication path, then the overt sender could also employ a lightweight signature scheme on the data stream to prevent modifications, which however is less likely if we assume that it cannot afford the computational power to encrypt the payload (see above). Yet if CS has compromised the sensor, then it can perform modifications before the payload is signed.

5 Detectability Evaluation

5.1 Experimental Setup

In order to determine detectability, we have computed[2] several metrics which may indicate that the sensor data stream has been tampered with: Shannon entropy [Edgar and Manz, 2017], compressibility [Cabuk et al., 2009], and bi-gram distribution [Kahn, 1996]. A random variable's entropy is a measure for the amount of information conveyed on average when its value is determined in a random trial. It is computed as

$$-\sum_k p_k \log_2 p_k$$

where k iterates over the random variable's possible values, p_k being the probability of the k-th value occurring as the outcome of a random trial. In our context, entropy can be employed as a detectability metric by comparing the entropy of the original unmodified data stream to the entropy of the data stream featuring an embedded covert channel: any discrepancy may be taken as an indication that information is being secretly transmitted. This reasoning applies to the two other metrics we have considered here in the same manner. Compressibility is computed by compressing 1000 consecutive elements in the data stream with `gzip` and dividing the original size by the compressed size. This is done 100 times with an offset increasing by 10 elements each time, starting at the beginning of

[2] The code is publicly available at https://github.com/sglitzinger/ccarchive.

the stream. In order to obtain the bi-gram distribution, we have determined the absolute frequencies of all ordered pairs of last l binary digit values over every two successive elements in the data stream. In our experiments, we have opted for $l \in \{1, 2, 4\}$.

Since the covert channel presented in Section 3.2 introduces an error, we will need to employ a metric to quantify said error in our experiments. The mean absolute percentage error (MAPE) [Myttenaere et al., 2016] yields the deviation of the value actually transmitted t from the original sensor datum d relative to the sensor datum and averaged over all n elements in the data stream:

$$\text{MAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{d_i - t_i}{d_i} \right| .$$

In the context of the experiments, covert channel bandwidth will also be considered. For our purposes, the number of sensor values used to transmit secret message bits relative to the total number of sensor values in the data stream is of primary interest and shall be denoted by ω.

As an example data set for our experiments we have chosen the "Gas sensor array temperature modulation Data Set" [Burgues et al., 2018, Burgues and Marco, 2018] from the UCI Machine Learning Repository [Dua and Graff, 2017]. It consists of data collected from 14 gas sensors in a gas chamber when being exposed to varying quantities of carbon monoxide and humid synthetic air. In addition to the sensors' time series data, carbon monoxide concentration, relative humidity, temperature, and flow rate are provided. In total, the set comprises data over a period of 13 days with a sample rate of $3.5\,\text{Hz}$. All values are assumed to be transmitted in the single-precision floating-point format according to the IEEE 754 standard. For each quantity and day, we have determined the distribution of sensor values. This enables us to identify similarities and choose a subset of sensor data for our experiments representing different distributions: gas sensors 1, 4, 9 delivering resistance values in megohm (MΩ), i.e., $10^6\,\Omega$, and the flow rate measured in milliliter per minute (mL/min), i.e., $10^{-3}\,\text{L/min}$ [Burgues et al., 2018].

In the following, we will focus on the data of gas sensor 1 (R1) and flow rate as the other distributions might necessitate different choices of parameters, but initial experiments have indicated that they do not deviate much in detectability.

5.2 Experimental Results

In our experiments, we are mainly concerned with two issues: the error introduced by altering sensor data to transmit the secret message and the effect this procedure has on detectability. Both can vary dramatically depending on a sensor's resolution. If there is a number for which a representation exists in between every two consecutive potential sensor values, the error can be kept low. Otherwise, the error very much depends on the distribution of actually transmitted values over the range of numbers which can be represented in the given format. In this context, the period λ over which sensor data is collected prior to establishing the covert channel may strongly influence error and detectability, and consequently was subject to our experimental evaluation. This holds true as well for the maximum absolute deviation θ from the original value one allows in case it must be modified by the covert sender when transmitting a bit of the secret message.

The frequent occurrence of a single value in the data stream may constitute an anomaly and thus create a pronounced negative effect on detectability, especially since

it is recognized by various metrics, e.g., bigram distribution. There are multiple reasons why a particular value v should occur frequently in the data stream featuring our covert channel:

- the sensor reading frequently yields v, and v is left unmodified sufficiently often in the transmission of the secret message,
- the set of previously collected values A contains few values, which means that there are values $v \in A$ many sensor values are mapped to if a `0` of the secret message is to be transmitted, and θ is sufficiently large,
- A contains many consecutive values, leading to a multitude of sensor values being mapped to a particular $v \notin A$ in the vicinity of these values if a `1` of the secret message is to be transmitted, and θ is sufficiently large.

The first case can be seen as unproblematic as it will not be perceived as an anomaly. The other two cases result from archiving sensor readings over a period of time that is either too short or too long in conjunction with a large θ. Simply lowering θ may however lead to a low covert channel utilization ω, i.e., a low fraction of sensor values serving as a vehicle to transmit secret message bits. Thus, the challenge is to identify a combination of λ and θ values resulting in a low error, a high ω, and inconspicuous detectability metrics. In our experiments, we have set $\theta = 10^i,\ i = -3, \ldots, 0$ and $\lambda = 10^i\,\mathrm{d},\ i = -4, \ldots, 1$, which leaves us with three days of data from the data set described in Section 5.1 to perform experiments with: we have implemented the covert channel based on the examined θ and the archives created over the various λ, and we have subsequently computed ω, MAPE, and detectability metrics on the resulting data streams.

Let us now consider the data for the R1 gas sensor on 13 October 2016 as an example. Implementing the covert channel for the various λ and θ values leads to the errors, relative entropies, and covert channel utilization figures ω displayed in Table 1. Unsurprisingly, a low error can be achieved by small λ and θ, which also means that bandwidth suffers tremendously, e.g., only 1.3% of all elements in the data stream serve to transmit secret message bits for $\lambda = 10^{-4}$ and $\theta = 10^{-3}$, cf. Table 1. Promising values for λ and θ are those where the error is low, and both relative entropy and ω are close to 1. A good candidate seems to be, e.g., $\lambda = 10^{-2}$, $\theta = 10^{-3}$: Here, the error is 0.028% on average, relative entropy amounts to 1.035, and we have $\omega = 0.653$, i.e., 65.3% utilization. Another option with higher ω would be $\lambda = 10^{-1}$, $\theta = 10^{-2}$, which yields >99% utilization, a lower error (0.024%) but a larger relative entropy (1.084). Figure 5 shows bigram distributions and compressibility values, respectively, for the latter case. Although these metrics do change, it is by no means obvious that a covert channel has been implemented.

Naturally, one will be interested in whether a particular choice of λ and θ can be maintained in the face of new data or whether one will have to reevaluate. Of course, this heavily depends on the concrete scenario and the nature of the data being transmitted. In our case, $\lambda = 10^{-1}$, $\theta = 10^{-2}$ serve pretty well on the two subsequent days; all considered quantities exhibit marginal changes only. The exact values for both 14/10/2016 and 16/10/2016 are:

- MAPE: 0.023,
- relative entropy: 1.087,

	MAPE				rel. entropy				cc util. ω			
	max. dev. threshold θ				max. dev. threshold θ				max. dev. threshold θ			
λ (d)	10^{-3}	10^{-2}	10^{-1}	1	10^{-3}	10^{-2}	10^{-1}	1	10^{-3}	10^{-2}	10^{-1}	1
10^{-4}	0.000	0.001	0.295	29.018	1.002	1.001	0.989	0.737	**0.013**	0.017	0.072	0.862
10^{-3}	0.032	0.116	0.672	0.870	1.000	0.971	0.929	0.908	0.320	0.524	0.729	0.937
10^{-2}	**0.028**	0.094	0.129	0.135	**1.035**	1.015	1.007	1.007	**0.653**	0.840	0.944	0.964
10^{-1}	0.023	**0.024**	0.024	0.025	1.084	**1.084**	1.084	1.084	0.971	**0.991**	0.992	0.993
1	0.023	0.023	0.023	0.023	1.091	1.091	1.091	1.091	1.000	1.000	1.000	1.000
10	0.023	0.023	0.023	0.023	1.091	1.091	1.091	1.091	1.000	1.000	1.000	1.000

Table 1: MAPE, relative entropy, and covert channel utilization for gas sensor R1, 13/10/2016

– $\omega = 0.975$.

It is conceivable that the procedure we have carried out manually could be automated as, essentially, we are confronted with an optimization problem: depending on the requirements in a given situation we can formulate constraints on error, utilization, or detectability, and subsequently pursue optimal parameter values for λ and θ w. r. t. a criterion (or multiple criteria) of our choice. Conceptually, one only needs similarity measures for bigram distribution and compressibility as well as a suitable optimization technique. To perform the optimization in practice, one must then specify constraints and—when optimizing for multiple objectives—weights to quantify each objective's relevance.

In its active phase, the archive-based covert channel can transmit up to one bit of the secret message per sensor value, ultimately depending on ω. Table 2 shows bandwidths for other covert channel approaches in the literature. [Mileva et al., 2021] present numerous covert channels in MQTT, two of which are experimentally evaluated. I4 encodes a secret message by publishing on MQTT topics, which have been chosen from a predefined set of topics and corresponding bit sequences. Thus, bandwidth depends on the cardinality of the set of potential topics. Initiating a reconnection by duplicating another client's identifier is the basic mechanism of covert channel I5. Each client is associated with a symbol[3] or bit sequence, which means that the covert receiver can piece together the secret message by observing series of reconnection actions. Consequently, the covert channel's bandwidth is determined by the number of client identifiers involved. In any case, the effective bandwidth depends on the frequency of the respective actions, which may be subject to constraints, either of a technical nature or with respect to detectability. Generally, other characteristics of a covert channel may be equally relevant to its quality as bandwidth, e.g., detectability[4], the computational effort required for sustained operation, difficulty of implementation, or applicability.

[3] In their analysis, the authors claim a bandwidth of 16 bit per reconnection as a single Unicode character is transferred [Mileva et al., 2021, p. 13]. Still, the maximum effective amount of information conveyed per single action obviously depends on the (size of the) set of symbols which may in fact occur.

[4] For instance, the authors qualify covert channel I5 as "easily detectable" [Mileva et al., 2021, p. 18].

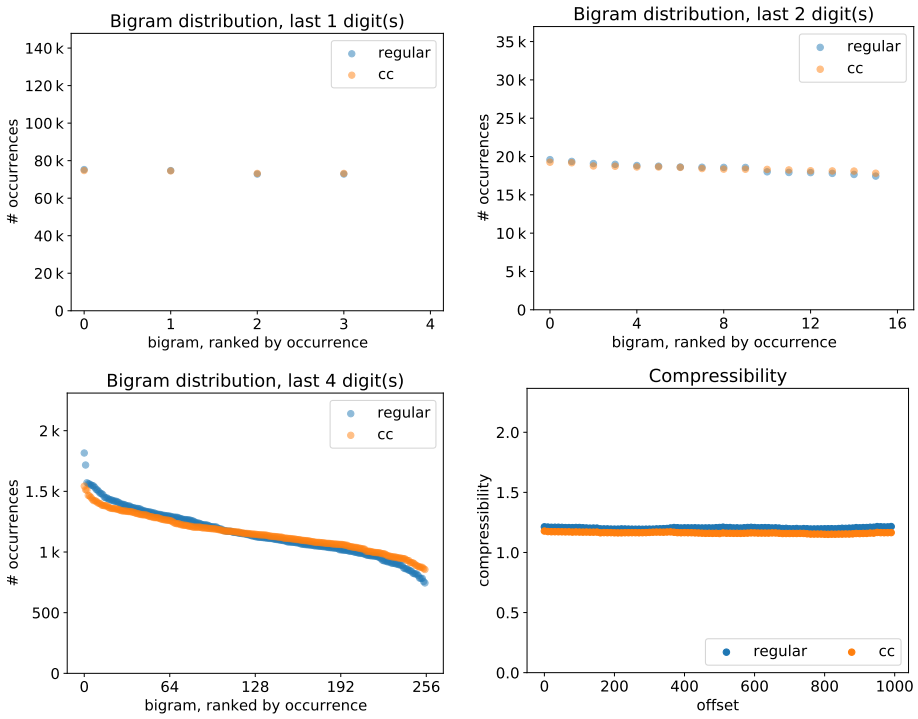

Figure 5: Bigram distribution and compressibility for the gas sensor R1 data stream on 13/10/2016 with and w/o covert channel implemented, $\lambda = 0.1\,\mathrm{d}$, $\theta = 10^{-2}$.

approach	action	bit/action
proposed here	send sensor value	≤ 1
I4 in [Mileva et al., 2021]	publish on MQTT topic	$\log_2$ #topics
I5 in [Mileva et al., 2021]	reconnection between MQTT client & broker	$\log_2$ #clients

Table 2: Covert channel bandwidths for various approaches in the literature

5.3 Practical Viability

In order to evaluate the practical viability of the covert channel approach presented in Section 3, we shall consider the additional computational effort caused by establishing the covert channel as well as its memory requirements.

We assume that the archive is kept in the shared memory where each entry occupies $4\,\mathrm{B}$ of memory for the value to be archived and an additional $8\,\mathrm{B}$ for the closest values not contained in the archive. If we do not statically compute the latter, we can make do with $4\,\mathrm{B}$ per entry in total, at the expense of higher computational costs at runtime to dynamically determine the closest values not in the archive whenever necessary. Depending on the values to be transmitted, less than $4\,\mathrm{B}$ per value may suffice (e.g., in case of half precision). Table 3 shows the memory requirements for two sensors from the dataset introduced in Section 5.1 and various data collection periods λ. These values

λ (d)	λ (s)	flow rate	R1
10^{-4}	8.64	0.348	0.348
10^{-3}	86.40	3.480	3.360
10^{-2}	864.00	27.996	22.764
10^{-1}	8,640.00	78.432	80.016
1	86,400.00	136.680	102.288
10	864,000.00	277.020	104.808

*Table 3: Memory Requirements (*kB*) for the archive over various collection periods*

assume that each entry occupies 12 B. If the closest values not in the archive are not statically precomputed, the memory requirements only amount to 1/3 of these figures. Initially, the memory requirements increase roughly linearly with growing λ. The more values the archive contains, the more likely it is that a particular sensor reading yields a value already present in the archive. Consequently, at some point, memory requirements begin to rise more slowly. For the R1 sensor and $\lambda = 10$, the remaining nine days after the first one lead to an increase in archive size of only 2.52 kB. The recommended data collection period of $\lambda = 10^{-1}$ d, cf. Section 5.2, brings about an archive size of ≈ 80 kB at 4 B per value, including the precomputed closest values not in the archive for each archive entry.

As a metric for the computational effort we opt for the CPU time, which comprises user and system time. consumed by the respective process. To this end, we have implemented the overt receiver on a system featuring an Intel Core i5-6600 and the overt sender on a Raspberry Pi 4 Model B, which could conceivably be a device performing sensor readings subsequently forwarded to a more potent system for further processing. The overt sender acquires sensor values from the shared memory at a rate of 3.5 Hz and immediately sends each value to the overt receiver via TCP. The covert sender is implemented to run in the same process as the overt sender, which enables us to determine the additional computational effort caused by establishing the covert channel in a straightforward manner: we can simply compare the CPU times for the scenarios with and without an active covert channel. For both approaches, we have collected CPU times for various numbers of transmitted values $\eta = 10^i, i \in \{2, 3, 4\}$, cf. Table 4. The data collection period was set to η values as well.

We have gathered CPU times for three individual phases:

1. building the archive by logging the transmitted sensor values (archiving phase),
2. statically precomputing the closest values not in the archive for each archived sensor value (mapping phase),
3. modifying the transmitted sensor values to convey the secret message (active phase).

In the archiving phase, we have overheads of -15.8 % to 1.1 %, which will likely not serve as a criterion to identify an attack due to the variance of CPU times present during regular operation. In the active phase of the covert channel, overheads amount to -9.2 % to 8.3 %. If a substantial number of values is transmitted, i. e., for $\eta = 10^4$, overheads are 1.4 % and 8.3 % for the resistance and flow rate sensor, respectively.

In the mapping phase, a mapping of archived values to closest values not in the archive is produced. The necessary computations commence after the archiving phase

is concluded, and immediately afterwards, i. e., with the subsequent sensor value, the active phase is entered, which means that the time period available for computing the mapping is somewhat short (it can be arbitrarily extended though if covert sender and covert receiver have agreed on that in advance). Fortunately, the computational effort required does not increase linearly with growing η, cf. Table 4, so large λ values are feasible. The reason for this behavior lies in the mapping algorithm where the archive is processed in ascending order, and the closest values not in the archive are carried along and updated when necessary. There is no need to actively search the closest values not in the archive for each archive entry, but a few simple comparisons suffice. In fact, CPU time consumed for computing the mapping remains nearly constant when going from $\eta = 10^2$ to $\eta = 10^3$ for the resistance sensor, and moving on to $\eta = 10^4$ incurs an $\approx 18\,\%$ increase in CPU time. A similar behavior can be observed for the flow rate, but the growth in CPU time amounts to $\approx 61\,\%$ when comparing $\eta = 10^3$ to $\eta = 10^4$. It can be expected that at some point, i. e., for some η, CPU time will not increase any further as no additional entries are stored in the archive due to the sensor readings not yielding any values so far unseen. The development of archive sizes over various λ in Table 3 supports this conjecture.

We consider a sender-side single-board computer such as a Raspberry Pi 4B as weak compared to a potent x86 CPU, which we expect on the receiver side. Although it is perfectly conceivable that such a single-board computer performs sensor readings and forwards the results, even weaker devices such as microcontrollers surely are an attractive option as well for this purpose. Relevant differences in our scenario lie in the processing power and in the amount of available memory. Whereas the Raspberry Pi 4B employed in our experiments features 2 GB of main memory, common microcontrollers are equipped with considerably less RAM. For instance, the Raspberry Pi Pico is provided with 264 kB [Raspberry Pi Ltd, 2023] and the Espressif ESP32 with 520 kB [Espressif Systems, 2023]. For most sensors and λ values, these amounts of memory should suffice, cf. Table 3. However, a large fraction of the device's memory may be occupied by the archive. In these situations, the archive could be kept, e. g., in the on-board flash or on memory cards (if available). Some devices may also have been upgraded with external RAM. In any case, memory requirements can be influenced via adequate parameterization. If need be, the size of the archive can be further reduced by computing the closest values not in the archive at runtime—at the expense of an additional computational effort.

The limited processing power of a microcontroller in comparison to a single-board computer is unlikely to constitute a serious issue: in our experiments, a transmission rate of 3.5 Hz implies that a sensor value is processed every 286 ms. As one can gather from Table 4, even in the most computation-intensive phase (the mapping phase), the per-value CPU time averages about 2.2 ms under the most unfavorable parameterization, i. e., 0.8 % of the interval between two successive sensor values on a single core of the Raspberry Pi 4B. It can thus be expected that a microcontroller will also be able to handle the required computations without overly high load values, especially given the fact that one would probably seek to optimize the code for the respective device.

We have not performed any measurements on the receiver side as we assume that any activity on the sender side will generally be more conspicuous due to the weaker system and presumably lighter cumulative computational load. The receiver's system on the other hand is likely to handle many different connections and associated computations, making it easier to hide any clandestine actions from an analysis of CPU and memory usage.

scenario	η	CPU time			
		flow rate		R1	
		total	per value	total	per value
regular transmission	10^2	38	0.381	36	0.364
	10^3	437	0.437	376	0.376
	10^4	3669	0.367	3990	0.399
CC (archiving)	10^2	36	0.363	32	0.324
	10^3	368	0.368	365	0.365
	10^4	3708	0.371	3705	0.371
CC (mapping)	10^2	100	0.995	224	2.241
	10^3	104	0.104	228	0.228
	10^4	167	0.017	270	0.027
CC (active)	10^2	36	0.359	35	0.355
	10^3	397	0.397	393	0.393
	10^4	3975	0.398	4047	0.405

Table 4: CPU time (ms) for regular sensor value transmission and all phases of the proposed covert channel, two sensors, and various counts of transmitted values

6 Conclusions

We have presented a covert channel to transmit secret data hidden in a stream of sensor data. The approach builds an archive of past sensor values and capitalizes on this knowledge shared by CS and CR to transmit secret message bits. Varying the length of the initial data collection period and the maximum deviation from a genuine sensor value tolerated allow for balancing error, detectability, and bandwidth.

We have discussed possible countermeasures to limit such a channel and possible ways to detect the presence of such a channel. In our application scenario MQTT, detection is difficult as floating-point data formats are not known to MQTT, thus these data are transmitted as binary payload data, and checking for correctness on the communication path would necessitate knowledge of the specific application that uses MQTT, and the data formats it applies. Still, the errors introduced by our approach lead to differences in entropy and can be detected, if each data value is used to transmit a bit of the secret message. Using only part of the transmitted values reduces detectability and steganographic bandwidth and thus presents the usual trade-off.

Future work will comprise devising an automated optimization approach for our archive-based covert channel to tune the λ and θ parameters and thus tailor the covert channel to specific application scenarios. Moreover, the feasibility of the covert channel for other types of payload data will be investigated. Finally, the representation of values that have been seen in a first phase, and finding corresponding values in close distance that have not been seen in the first phase may be an application of machine learning. Thus, future work will also comprise the application of machine learning algorithms to our problem.

References

[Banks et al., 2019] Banks, A., Briggs, E., Borgendale, K., and Gupta, R. (2019). Mqtt v5.0. OASIS Std.

[Becher et al., 2006] Becher, A., Benenson, Z., and Dornseif, M. (2006). Tampering with motes: Real-world physical attacks on wireless sensor networks. In Clark, J. A., Paige, R. F., Polack, F., and Brooke, P. J., editors, *Security in Pervasive Computing, Third International Conference, SPC 2006, York, UK, April 18-21, 2006, Proceedings*, volume 3934 of *Lecture Notes in Computer Science*, pages 104–118. Springer.

[Burgues et al., 2018] Burgues, J., Jimenez-Soto, J. M., and Marco, S. (2018). Estimation of the limit of detection in semiconductor gas sensors through linearized calibration models. *Analytica chimica acta*, 1013:13–25.

[Burgues and Marco, 2018] Burgues, J. and Marco, S. (2018). Multivariate estimation of the limit of detection by orthogonal partial least squares in temperature-modulated MOX sensors. *Analytica chimica acta*, 1019:49–64.

[Cabuk et al., 2009] Cabuk, S., Brodley, C. E., and Shields, C. (2009). IP covert channel detection. *ACM Transactions on Information and System Security (TISSEC)*, 12(4):22:1–22:29.

[Dua and Graff, 2017] Dua, D. and Graff, C. (2017). UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences.

[Edgar and Manz, 2017] Edgar, T. W. and Manz, D. O. (2017). Chapter 2 - science and cyber security. In Edgar, T. W. and Manz, D. O., editors, *Research Methods for Cyber Security*, pages 33–62. Syngress, Amsterdam.

[Espressif Systems, 2023] Espressif Systems (2023). *ESP32 Series Datasheet*.

[Fridrich, 2009] Fridrich, J. (2009). *Steganography in Digital Media: Principles, Algorithms, and Applications*. Cambridge University Press, Cambridge, UK.

[Heßeling, 2023] Heßeling, C. (2023). *Network Storage Covert Channels in Sensor Data Transmissions based on Recoding IEEE 754-compliant Number Representations*. PhD thesis, FernUniversität in Hagen, Hagen.

[Heßeling et al., 2022] Heßeling, C., Keller, J., and Litzinger, S. (2022). Network steganography through redundancy in higher-radix floating-point representations. In *ARES 2022: The 17th International Conference on Availability, Reliability and Security, Vienna,Austria, August 23 - 26, 2022*, pages 48:1–48:7. ACM.

[Heßeling and Keller, 2022] Heßeling, C. and Keller, J. (2022). Pareto-optimal covert channels in sensor data transmission. In *EICC 2022: Proccedings of the European Interdisciplinary Cybersecurity Conference*, EICC 2022, pages 79–84, New York, NY, USA. Association for Computing Machinery.

[Ho, 2019] Ho, J. (2019). Covert channel establishment through the dynamic adaptation of the sequential probability ratio test to sensor data in iot. *IEEE Access*, 7:146093–146107.

[Hou et al., 2023] Hou, N., Xia, X., and Zheng, Y. (2023). Cloaklora: A covert channel over lora PHY. *IEEE/ACM Trans. Netw.*, 31(3):1159–1172.

[Jankowski et al., 2013] Jankowski, B., Mazurczyk, W., and Szczypiorski, K. (2013). PadSteg: Introducing inter-protocol steganography. *Telecommun. Syst.*, 52(2):1101–1111.

[Kahn, 1996] Kahn, D. L. (1996). *The codebreakers: the story of secret writing*. Scribner, New York, NY.

[Lampson, 1973] Lampson, B. (1973). A note on the confinement problem. *Communications of the ACM*, 16(10):613–615.

[Li et al., 2022] Li, Z., Chen, B., Chen, X., Li, H., Xu, C., Lin, F., Lu, C. X., Ren, K., and Xu, W. (2022). Spiralspy: Exploring a stealthy and practical covert channel to attack air-gapped computing devices via mmwave sensing. In *29th Annual Network and Distributed System Security Symposium, NDSS 2022, San Diego, California, USA, April 24-28, 2022*. The Internet Society.

[Marpe et al., 2003] Marpe, D., Wiegand, T., and Schwarz, H. (2003). Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. *IEEE Trans. Circuits Syst. Video Technol.*, 13(7):620–636.

[Mazurczyk et al., 2014] Mazurczyk, W., Szaga, P., and Szczypiorski, K. (2014). Using transcoding for hidden communication in IP telephony. *Multim. Tools Appl.*, 70(3):2139–2165.

[Mazurczyk et al., 2018] Mazurczyk, W., Wendzel, S., and Cabaj, K. (2018). Towards deriving insights into data hiding methods using pattern-based approach. In Doerr, S., Fischer, M., Schrittwieser, S., and Herrmann, D., editors, *Proceedings of the 13th International Conference on Availability, Reliability and Security, ARES 2018, Hamburg, Germany, August 27-30, 2018*, pages 10:1–10:10, New York, NY. ACM.

[Mazurczyk et al., 2016] Mazurczyk, W., Wendzel, S., Zander, S., Houmansadr, A., and Szczypiorski, K. (2016). *Information Hiding in Communication Networks: Fundamentals, Mechanisms, and Applications*. IEEE Series on Information and Communication Networks Security. Wiley, New York, NY.

[Menezes et al., 1996] Menezes, A. J., van Oorschot, P. C., and Vanstone, S. A. (1996). *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL.

[Mileva et al., 2021] Mileva, A., Velinov, A., Hartmann, L., Wendzel, S., and Mazurczyk, W. (2021). Comprehensive analysis of MQTT 5.0 susceptibility to network covert channels. *Computers & Security*, 104:102207.

[Myttenaere et al., 2016] Myttenaere, A. D., Golden, B., Grand, B. L., and Rossi, F. (2016). Mean absolute percentage error for regression models. *Neurocomputing*, 192:38–48.

[Patuck and Hernandez-Castro, 2013] Patuck, R. and Hernandez-Castro, J. (2013). Steganography using the extensible messaging and presence protocol (XMPP). *CoRR*, abs/1310.0524.

[Pigeon and Bengio, 1998] Pigeon, S. and Bengio, Y. (1998). A memory-efficient adaptive huffman coding algorithm for very large sets of symbols. In *Data Compression Conference, DCC 1998, Snowbird, Utah, USA, March 30 - April 1, 1998*, page 568. IEEE Computer Society.

[Rajba et al., 2023] Rajba, P., Keller, J., and Mazurczyk, W. (2023). Proof-of-work based new encoding scheme for information hiding purposes. In *Proc. 18th International Conference on Availability, Reliability and Security (ARES 2023), Benevento, August 29-Sept 1, 2023*, New York, NY. ACM.

[Ramsden, 2015] Ramsden, J. (2015). *Bioinformatics - An Introduction, Third Edition*, volume 21 of *Computational Biology*. Springer.

[Raspberry Pi Ltd, 2023] Raspberry Pi Ltd (2023). *Raspberry Pi Pico Datasheet*.

[Santos et al., 2018] Santos, P. M., Rodrigues, J. G. P., Cruz, S. B., Lourenço, T., d'Orey, P. M., Luis, Y., Rocha, C., Sousa, S., Crisóstomo, S., Queirós, C., Sargento, S., Aguiar, A., and Barros, J. (2018). PortoLivingLab: An IoT-based sensing platform for smart cities. *IEEE Internet Things J.*, 5(2):523–532.

[Tuptuk and Hailes, 2015] Tuptuk, N. and Hailes, S. (2015). Covert channel attacks in pervasive computing. In *2015 IEEE International Conference on Pervasive Computing and Communications, PerCom 2015, St. Louis, MO, USA, 23-27 March, 2015*, pages 236–242. IEEE Computer Society.

[Vitter, 1987] Vitter, J. S. (1987). Design and analysis of dynamic Huffman codes. *J. ACM*, 34(4):825–845.

[Wang et al., 2006] Wang, Q., Zhu, Y., and Cheng, L. (2006). Reprogramming wireless sensor networks: challenges and approaches. *IEEE Network*, 20(3):48–55.

[Wendzel et al., 2021] Wendzel, S., Caviglione, L., Mazurczyk, W., Mileva, A., Dittmann, J., Krätzer, C., Lamshöft, K., Vielhauer, C., Hartmann, L., Keller, J., and Neubert, T. (2021). A revised taxonomy of steganography embedding patterns. In *ARES 2021: The 16th International Conference on Availability, Reliability and Security, Vienna, Austria, August 17-20, 2021*, pages 67:1–67:12, New York, NY. ACM.

[Yarochkin et al., 2008] Yarochkin, F. V., Dai, S., Lin, C., Huang, Y., and Kuo, S. (2008). Towards adaptive covert communication system. In *14th IEEE Pacific Rim International Symposium on Dependable Computing, PRDC 2008, 15-17 December 2008, Taipei, Taiwan*, pages 153–159. IEEE Computer Society.

[Zhang, 2022] Zhang, Y. (2022). *Mobile Edge Computing*, volume 9 of *Simula SpringerBriefs on Computing*. Springer, Cham.