Adaptable Hardware Fingerprinting for Radio Data Links and Avionics Buses in Adversarial Settings

Simon Birnbach*, Joshua Smailes*, Richard Baker*, Ivan Martinovic*

*University of Oxford
Oxford, United Kingdom
first.last@cs.ox.ac.uk

***Abstract*—Despite their substantially different purposes, a common issue both for many legacy data links and many onboard data buses used in aviation is a lack of authentication and thus a vulnerability to spoofing and message manipulation attacks. This problem has been discussed at length for some prominent technologies of both sorts (*e.g.*, ADS-B, ARINC 429) but is common in many more cases (*e.g.*, ACARS, CPDLC, MIL-STD-1553, RS-485). For ground data links, attacks can take place over-the-air, while for onboard buses an attacker requires some physical access to an aircraft. Yet, in both cases an attacker's goal is to obtain control of a transceiver on the communication channel. As such, hardware fingerprinting methods are useful in both contexts. Prior work has applied such methods to specific protocols. We now propose a transferable fingerprinting scheme that has applicability in both contexts, describe our experiences in applying it for each and evaluate its performance in benign and malicious conditions. As replacing legacy communication links with newer, more secure protocols is practically challenging, the improvements to practical deployment offered by our system represent a meaningful benefit for real deployment efforts.**

I. Introduction

Modern aircraft systems rely on a wide range of communications links, both onboard in avionics data buses and outside as radio data links. Alongside the need for these links to be reliable and robust, there is also an ever-growing need for security. In particular, most links of both classes currently in use are unauthenticated by nature. This type of vulnerability exists because it was previously assumed that attackers would lack the appropriate equipment and knowledge to interfere with these links—an assumption which is now breaking down. The problem of spoofing and message manipulation attacks has been discussed for a wide range of avionics data links and onboard buses, including ADS-B [12], ARINC 429 [13], ACARS [8], CPDLC [10], and MIL-STD-1553 [5].

While there are plenty of cryptographic solutions available, they are unlikely to be adopted in the short term due to the extensive certification processes enforced by legislation and safety procedures. To improve the longevity of the new solution, it is preferred to ultimately replace legacy systems with new and secure ones, rather than re-engineering existing systems. In the meantime, however, there is still a need for novel techniques to secure these systems without changing the protocols themselves.

To tackle this issue, various transmitter fingerprinting methods have been proposed by prior work [1, 3, 4]. However, existing approaches typically require substantial manual design and implementation effort and need to be extensively trained on the target deployment. In this paper, we examine methods to overcome these drawbacks and secure legacy communication links in a manner that is more flexible and easily-applied. We propose a machine-learning based method utilising Siamese networks, which can be quickly trained *in-situ* and offers the benefits of both automatic feature extraction and small training data sizes.

In this paper we describe a system that has been developed and successfully applied for the passive verification of both ADS-B data-link messages and RS-485 serial data bus communications. The system requires no changes to transmitters, but instead can simply be deployed in a single listener. Our system undertakes signal-level fingerprinting of the raw waveforms each link generates. By careful design for flexibility we adopt an approach that can adapt to the very different technologies and requires limited manual analysis effort to generalise further to other protocols and signal types.

In consideration of this, we employ machine learning methods due to their promising performance in related problems in this domain [6, 15]. Besides being of technical interest, these methods also support the applicability of the technology to many cases, simplify deployment without deep knowledge of the peculiarities of the physical hardware and permit adaptation to the properties of the communications link itself.

For radio data links, over ADS-B, we use a software-defined radio (SDR) chain to capture and decode raw signals (USRP B205-mini, dump1090). We collect a dataset from real aircraft spanning 11 months. In the RS-485 data bus case, we attach an off-the-shelf oscilloscope device (Picoscope 5244D MSO) and commonplace protocol decoding libraries. We capture a dataset with a range of transmitters of both equal and different makes. The system is trained on accessible hardware, without dedicated high-performance computing resources.

In both cases, we evaluate the system under benign conditions and in the presence of a range of attacks. These attacks consider compromised transmitters, newly-attached malicious ones and even a powerful attacker who can capture and mimic signals from other devices. We show that the system adapts, in the ADS-B case, to operate on aircraft that were unseen at training time. Meanwhile, in the RS-485 case, the more constrained problem allows it to deliver both reliable detection

TABLE I
COMPARISON OF RELATED WORK

Reference	ID masking	Transmitter set	Data collection	Over-the-air	On-the-bus
Gopalakrishnan et al. [2]	✔	Closed	Short-term	✔	✗
Ying et al. [14]	✗	Closed	Short-term	✔	✗
Leonardi et al. [3]	✗	Open	Medium-term	✔	✗
Zha et al. [15]	✗	Closed	Short-term	✔	✗
Nicolussi et al. [6]	✔	Closed	Medium-term	✔	✗
Levy et al. [4]	✗	Closed	Short-term	✗	✔
Stan et al. [11]	✗	Closed	Short-term	✗	✔
Onodueze et al. [7]	✗	Open	Short-term	✗	✔
Gilboa-Markevich et al. [1]	N/A	Closed	Short-term	✗	✔
Our work	✔	Open	Long-term	✔	✔

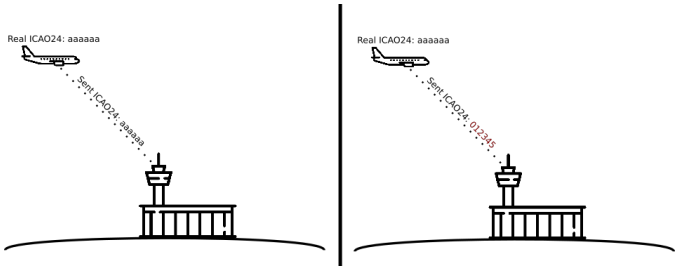

Fig. 1. Illustration of the attack model. In the left frame the aircraft transmits messages with its true ICAO24 identifier, in the right frame the behaviour and arrangement is identical except that the wrong identifier is sent.

of malicious devices and also low false-alarm rates—even when operating in noisy environments. We also describe our experience in making the required (minor) modifications to the system in order to apply it in each case.

II. RELATED WORK

In this section, we present related work that considers securing communication through transmitter fingerprinting techniques in the contexts of ADS-B radio links and avionics bus systems.

A. Fingerprinting of ADS-B transmitters

The authors of [2] use a convolutional neural network (CNN) to generate and verify transmitter fingerprints for both WiFi and ADS-B transmitters, increasing robustness by subjecting the data to synthetic noise. They observe that artificially high accuracy is achieved if the entire message is used for the fingerprint due to the model extracting identifiers from the message, and recommend to only rely on the preamble for fingerprinting. We address this issue in our work by masking the identifying information contained in the message before passing it onto the fingerprinter.

In [6] the authors also use a CNN, classifying transmitters based on magnitude and phase information contained within message preambles. This does not require masking, since the preamble does not contain any aircraft identifiers.

The authors of [14] design an ADS-B spoofing detector, using two deep neural networks: a message classifier to detect ground-based attackers, and an aircraft classifier to distinguish individual aircraft transmitters. Similarly, in [15] the authors classify transmitters by transforming IQ samples into a 2-dimensional pictorial representation, and applying well-known image classification techniques to these representations.

Finally, the authors of [3] build an ADS-B intrusion detection system, comparing each incoming message to historic messages and using statistical analysis to test if the new message is sufficiently similar to past examples. This allows new aircraft to be added to the system without requiring model retraining.

These works have made significant contributions in the field of ADS-B fingerprinting, but many of them share a variety of shortcomings. Not all of these systems ensure the transmitter identifier is removed from the classifier, making them vulnerable to identifier spoofing attacks. Moreover, most of these systems use classifiers which can only identify a closed set of aircraft. This does not scale to the requirements of modern air traffic control systems, which deal with thousands of aeroplanes per day—many of which are seen for the first time, or only a handful of times before. Finally, none of the papers evaluate their system over longer periods of time. This casts doubt on the time stability of fingerprints and their transferability between different locations. For a comparison of the presented systems, please refer to Table I.

B. Fingerprinting of transmitters on avionics buses

There is less work looking at fingerprinting communications over common avionics buses. Unlike ADS-B, these use physical wires to facilitate communication between devices on an aircraft. They suffer from vulnerabilities to spoofing by malicious devices communicating over the shared bus.

The authors of [4] detect malicious devices on the MIL-STD-1553 avionics bus by using a CNN to classify voltage signals associated with a claimed sender as legitimate or not. They also detect new devices connected to the bus by training an autoencoder to detect when voltage patterns have changed, and detect anomalies and manipulated messages using another autoencoder. Similarly, the system in [11] uses a clustering algorithm to build profiles for transmitters on the MIL-STD-1553 bus to identify devices, and performs anomaly detection by looking for invalid message sequences. Finally, the authors of [7] assess a number of classification and anomaly detection algorithms on the same dataset as [11], focusing on classifying attacks and benign communication rather than authenticating specific transmitters.

In [1], devices on the ARINC 429 bus are authenticated by sampling messages at 50 MS/s. For each transmitter on the bus, signals corresponding to every sequence of bytes is gathered, and their characteristics are compared using PCA.

Each of these papers requires message data to be gathered for each classified transmitter on the bus to measure device characteristics, resulting in additional work each time a device is added to the bus. This differs from our work which, once trained, can be deployed and used to authenticate an open set of transmitters, requiring only a small number of messages from the new transmitter to start authenticating its messages—these can be gathered automatically as the device is added.

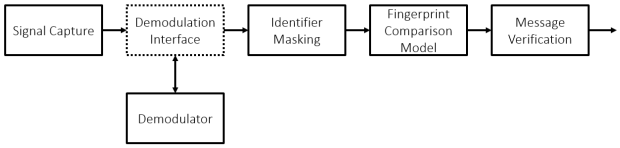

Fig. 2. Overview of the physical-layer fingerprint verification system.

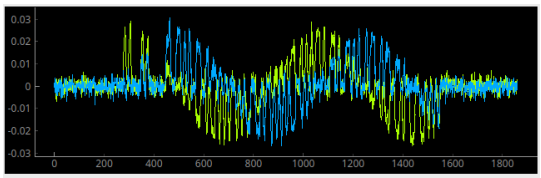

Fig. 3. Example raw waveform, for an ADS-B position message, captured using 2-channel IQ sampling. The in-phase (I) channel is shown in green and the quadrature (Q) channel in blue.

III. Threat Model

We consider an attacker attempting to masquerade as a victim transmitter. This may be in the presence or absence of the victim. We consider two distinct forms of this attack:

a) Weak Attacker: impersonates victim at a data level, but is incapable of mimicking the physical-layer behaviour of the victim. This is applicable to situations in which a benign transmitter is compromised remotely, or in which a new off-the-shelf transmitter has been introduced by the attacker.

b) Strong Attacker: has full control of the physical-layer functionality of their transmitter and can take steps to mimic a victim up to arbitrary precision bounded by cost and available equipment. This is applicable to situations in which an attacker deploys their own equipment to conduct an attack.

This threat model is applicable to both radio data links and onboard data buses, and we expect our system can be in either case. As an example of how such an attack might be executed, consider ADS-B position reports. In this case the attacker attempts to sent position reports that identify them as a different aircraft. The *weak attacker* does this by simply modifying the ICAO24 identifier they send (*cf.,* Figure 1), whereas the *strong attacker* additionally can additionally attempt to change their physical transmission signature.

To remain conservative in our assumptions, we consider the attacker to construct completely valid messages, transmitted in the correct airborne location and at appropriate times in the case of ADS-B, but with a claimed identifier that does not match the actual transmitter. In reality an attacker would likely falsify other values as well, which may make them easier to detect, but we do not include such behaviour in this study, to prevent it from falsely improving our results. On the contrary, in the case of our RS-485 study, we intentionally omit identifiers from the message format in order to increase the difficulty of the problem. The attacker is therefore not attempting to impersonate only one victim (a verification task), but instead attempting to be accepted as any victim (a harder identification and verification task).

IV. System Design

This section describes the system at a conceptual level, illustrated by an ADS-B application, but with general applicability. In Sections V-A and V-B we detail the specifics of the implementation used for ADS-B and RS-485 verification.

A. System Overview

A high-level illustration of the system is given in Figure 2. It consists of five stages: signal capture, demodulation, masking, fingerprinting and message verification.

A signal waveform is first captured from a receiver. This can conceptually be any transmitted waveform that can be suitably demodulated and decoded into a corresponding message. We illustrate an example message, for an ADS-B position, in Figure 3. The captured wave can be processed to extract the intended message, but is also retained in a raw form from which additional information about the transmitter identity can be extracted. Note that the waveform is captured by simple message detection triggers and may not be selected precisely in time. Figure 3 shows this, with the waveform containing periods of empty noise before and after the message itself.

The waveform is then passed to a demodulation and decoding stage, which is responsible for extracting the encoded message from the waveform. This stage can either be included as part of the system or use an existing message reception capability. The simplest example is an SDR for messages that can be interfaced with to pass raw waveforms to. Such receivers are commonplace in crowdsourced monitoring and are becoming progressively more popular in commercial operations. The waveform is provided to the demodulator via a suitable interface, depending on the demodulator implementation.

The extraction process performed by the demodulator provides both the final message, which will later be verified, but also necessarily involves identifying precisely where in the waveform duration the message is encoded. While the waveform may have been captured initially with rough time synchronisation and some uncertainty of whether it was a valid message (as opposed to interference or noise), at this stage the demodulator provides a far more accurate synchronisation and a confirmation that the message is indeed valid.

The message data and timings are read back by the system, from the demodulator. The timings are used to subsample the original waveform to only that representing the message itself (see Figure 4), denoted henceforth as m and is passed further through the system, along with the extracted message data.

In the next stage, identifiers are masked from the waveform. In many protocols the identity of the transmitter is encoded directly in the message and thus has been modulated in the waveform m. This leads to a trivial distinction in the waveforms of transmitted messages of different aircraft, but one that is due to input data rather than hardware characteristics.

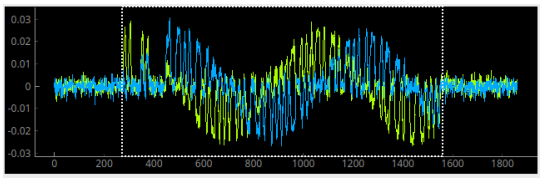

Fig. 4. Example waveform from Fig. 3 overlaid with dotted-line box denoting tight time synchronisation, as determined by the demodulator. The signal is subsequently truncated to this box and paired with the extracted message data.

The data could easily be influenced by an attacker. In order to avoid the fingerprint being affected by this field, it is masked at this stage. More precisely, the IQ samples are zeroed in the region of the waveform that corresponds to the identifier.

The masked waveform and the message data are, at this stage, available for comparison with other waveforms and therefore ready for fingerprinting. The intricacies of these stages are discussed in detail in the following sections, but we describe them at an overview level here.

The masked waveform is first passed through a machine-learning model that derives a fingerprint for the transmitter. The model has been trained to extract transmitter-specific detail and ignore message data, conditions or extraneous noise. The model should produce very similar fingerprints for messages from the same transmitter, even if they have different content and were sent at different times from different locations and with different environments. Conversely, messages from different transmitters that were sent at similar times, from nearby one another and with similar weather and propagation conditions, should still have distinct fingerprints. This fingerprint can be compared with those produced by other messages. The fingerprinter therefore extracts the claimed identity of a transmitter from the message data and uses it to select an appropriate comparison message(s). If the fingerprinter has not seen the claimed transmitter before then it will store the fingerprint of the first message for future testing, otherwise it will select fingerprints produced from one or more messages, from its previous observations, and compare the new message to them. If the fingerprint similarity is within an acceptable tolerance then the message is considered to be valid, otherwise it is flagged as suspicious.

B. Siamese Network

A Siamese network design permits a comparison between multiple inputs, by deriving a low-dimensional representation and then computing a distance metric. In this case, it takes two waveforms as input, produces a small intermediate vector for each, computes a distance metric between the two vectors and then derives a similarity score based on the distance metric. The similarity score is the final output.

The Siamese network is trained by comparing pairs of waveforms, both from the same transmitter and from different transmitters. A distinct advantage of this approach is that the network learns, in general terms, what factors indicate similarity or difference between transmitters. A fingerprint is not explicitly associated with any particular transmitter, it is simply a representation that is conditioned to be similar for the same transmitter and distinct otherwise.

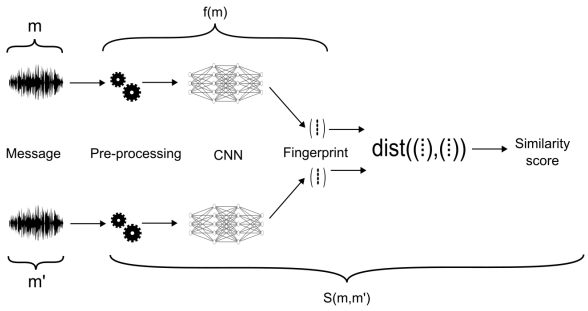

Fig. 5. Siamese network diagram.

This approach confers the benefit that the model can handle new, unseen transmitters when in use, even without being trained on them. What is important is that the training has allowed the model to observe all the types of variation that occur between different transmitters—even if it has not seen all the individual units. An additional benefit comes from the ability to take advantage of limited input data in a few-shot learning scenario. The available training data scales as N^2 where N is the number of input messages. Adding a single new message to the training set allows the model to then be trained by comparing the message to every existing one. Even with a small amount of data the model can exploit it well in order to identify fine variations that can form a fingerprint.

Within the Siamese architecture itself, a convolutional neural network (CNN) is used to extract the fingerprint. This structure is useful for its feature-extraction capabilities, long exploited in the computer vision field for object identification.

C. Fingerprinting and Fingerprint Verification

When a message sent with a previously unseen identifier has been received, the system has to build a fingerprint for this new transmitter. This fingerprint can be formed from this message only, or by incorporating future messages as well. In this work, we define three different fingerprinting methods: FIRST, BEST-OF-N, and CENTROID.

FIRST creates a fingerprint based on the first message received from a new transmitter. This method is simple and fast, requiring only a single message to form the fingerprint—it is thus ready to be used immediately after a new transmitter has been identified. However, this does not lead to robust fingerprints, since future verification performance is highly dependent on the quality of the message, and how representative it is of other messages from the same transmitter.

To tackle this challenge, several messages can be combined to form a more robust fingerprint. BEST-OF-N compares the first N messages received, creating a fingerprint from the message which is most similar to the other $N-1$ messages received. This is achieved by taking the message that maximises the sum of the similarity scores to the other messages. This method introduces an interesting trade-off, as

the system has to wait longer for additional messages to build a more robust fingerprint, and the computational complexity of finding the fingerprint scales quadratically with the number of messages N. However, this method reduces the risk of taking a fingerprint which is not representative of the transmitter.

A less computationally intensive method that still considers multiple messages is CENTROID, which computes the fingerprint by taking the mean of the message fingerprints of the first N received messages. The lower computational cost means that the system can potentially use a larger number of messages N to further increase robustness.

We formally define the fingerprinting methods as follows: Let $\mathcal{M} = \{m_1, m_2, .., m_N\}$ be the first N received messages from a new identifier. Then:

$$\text{FIRST} := f(m_1) \tag{1a}$$

$$\text{BEST-OF-N} := f\left(\arg\max_{m \in \mathcal{M}} \sum_{\substack{m' \in \mathcal{M}, \\ m' \neq m}} \mathcal{S}(m, m')\right) \tag{1b}$$

$$\text{CENTROID} := \frac{\sum_{m \in \mathcal{M}} f(m)}{|\mathcal{M}|} \tag{1c}$$

where $f(m)$ is the output at the last layer of the underlying CNN when the Siamese network is given message m as an input, and $\mathcal{S}(m, m')$ is the overall output by the Siamese network when given two messages. Intuitively, $f(m)$ is the fingerprint of the message m as derived by the CNN, whereas $\mathcal{S}(m, m')$ is the similarity score of the two messages m and m'. See Figure 5 for a visual representation of the different parts of the Siamese network. In practice $f(m)$ can be precomputed and stored for later use, so that only half the Siamese network has to be executed during live operation.

V. IMPLEMENTATION DETAILS

The system described in Section IV was implemented for verifying ADS-B messages and then subsequently for RS-485 communication. Specific instances of each component were made to support this and are described in this section.

In both cases, the components of the system were implemented with a range of technologies, discussed in individual sections below. Each component was packaged into a Docker container and all were connected with ZeroMQ PUB-SUB sockets. This architecture makes it simple to deploy any number and combination of components, as long as they conform to a fan-out feed-forward pattern. In the simplest case, this can be a linear arrangement following Figure 2 directly. However, more complicated arrangements with multiple subscribers are also available. The use of ZeroMQ also enables the components to be placed arbitrarily; either on a single host, in multiple containers, or distributed across the Internet. The full source code for the system implementation is available at: https://github.com/ssloxford/auto-phy-fingerprint.

A. ADS-B Processing

1) Signal Capture: A *GNURadio* flowgraph is used to capture ADS-B transmissions. The flowgraph operates an SDR (here, an Ettus USRP B205mini-i), tuned to the ADS-B downlink frequency of 1090 MHz and sampling at 20 MS/s. ADS-B has a bandwidth of 2 MHz, so this results in a $10\times$ oversampling rate. The radio's bandwidth filter is set to 3 MHz to remove high-frequency noise outside the region of interest.

The signal stream provided from the radio is filtered by a power trigger that drops periods with no transmission. This trigger does not depend on a particular power level—message power will vary substantially depending on the distance from the transmitter—instead, it is tuned to look for an increase in power corresponding to the approximate length characteristics of an ADS-B message. Even when configured conservatively, this substantially reduces the processing power required by the data collection system.

2) Demodulation: The demodulator was created from a modified version of the `libmodes` library[1], which was in turn built by restructuring the `dump1090` utility[2]. This library takes a waveform burst as input, and indicates the presence of an ADS-B message, as well as decoded message data and sample-accurate timing information. Our system samples at 20 MS/s, but `libmodes` assumes a 2 MS/s sample rate—as a result, our demodulator must produce a downsampled copy of the waveform, pass it to `libmodes`, then upscale the timing results to truncate the original waveform and pass it downstream to the next component. This results in output waveforms with an identical length (2400 samples) across two channels (I & Q), alongside 14 bytes of message data and additional message metadata.

3) Filtering: Mode-S and ADS-B carry a range of messages, but only position messages are of interest in this study. As a result, any other messages are filtered by an optional module, implemented as a separate stage to permit easy inclusion or removal from the pipeline as desired.

4) Masking: To prevent the fingerprinter from identifying transmitters based on message contents (rather than characteristics of the waveform), the ICAO24 aircraft identifier is masked out of the waveform. This is performed by a simple routine that removes the samples corresponding to the ICAO24, by setting them to zero. The ICAO24 is stored in bytes 3–5 of the ADS-B message, which corresponds to samples 320–800, when accounting for oversampling and the message preamble. The masking routine is called when the waveform is used and so takes place within the fingerprinter.

B. RS-485 Processing

Unlike ADS-B, which mandates a specific message format, RS-485 specifies only physical-layer characteristics and can carry any kind of message traffic. The minimum unit of transmission is a single byte and as such, for maximum generality in our work, we implemented the verification system such that it operated on individual bytes in RS-485.

1) Signal Capture: Signal capture is performed by a Python script controlling an oscilloscope (here, a Picoscope 5244D MSO). The oscilloscope is connected to the RS-485 bus on

[1] Available at https://github.com/ssloxford/libmodes

[2] Available at https://github.com/antirez/dump1090

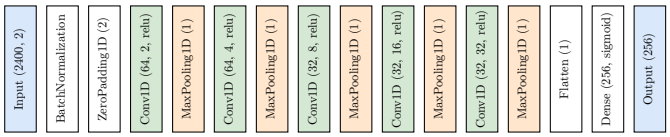

Fig. 6. The layers of the neural network comprising the model's fingerprint extractor. This is used to reduce each input waveform to its fingerprint.

both lines and wired to the shared ground, enabling a capture of the full differential signal. The oscilloscope is configured with a simple trigger that collects a signal once a voltage above the threshold for a '1' bit is exceeded, thus being triggered by the start bit of a byte. A capture is then performed for the duration of a single byte, as determined by the baud rate of the RS-485 link. At a low baud-rate this can lead to the signal being heavily oversampled even at modest capture rates (here, by 245×), but can operate with far lower rates if necessary.

2) Demodulation: Straightforward software-based decoding is used. In our case this was implemented as custom software, but such decoders are common in most signal capture tools or bus monitors. [3] The demodulator computes the differential signal by summing the two channels, averages the signal level across the time window for each bit and then compares it to a threshold to determine the value. Each resulting byte value is then output, along with information about its position in the original message.

3) Filtering & Masking: No filtering or masking are required in this case, as every byte is relevant to the system and no transmitter identifiers are sent. However, in other protocols this may not be the case. But as full waveforms are available and each byte is enumerated with its value, it is simple to filter based on any value or mask any sections containing identifiers.

C. Comparison Model

The Siamese model for fingerprint extraction and comparison was implemented in Python using Keras with the Tensorflow 2 backend. Once trained (this process is detailed for each dataset in Section VI) the model can be used as a component of the fingerprinter. It accepts two input vectors of suitable length for the communication technology (2400 for ADS-B, 8450 for RS-485) with 2 channels (I & Q) and outputs a single similarity score value.

Pre-processing layers are applied to normalise and pad the input, followed by a series of 5 convolutional layers, each followed by a max-pooling layer. The result is then flattened and coerced into a vector of length 256 by a single dense layer. These layers are shown in Figure 6.

Two inputs are passed through this network (with the same parameters and weights) to produce fingerprints for each input. These are then compared by calculating the L1 distance between them, followed by a final dense layer to produce a similarity score in the range $[0, 1]$.

[3] *e.g.*, https://www.picotech.com/library/oscilloscopes/serial-bus-decoding-protocol-analysis or https://www.sigrok.org/wiki/Protocol_decoder:Modbus

TABLE II
ACCURACY COMPARISON FOR EACH OF THE FINGERPRINTING METHODS.

	Overall Accuracy		Short-term Accuracy		Long-term Accuracy	
Method	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
First	0.636	0.421	0.744	0.388	0.623	0.422
Best of N	0.702	0.409	0.819	0.349	0.687	0.413
Centroid	0.678	0.420	0.804	0.364	0.663	0.424

D. Fingerprinter

Finally, the fingerprinter component coordinates the fingerprinting process. This includes model execution, fingerprint tracking, and verification output. It is implemented as a Python module, interfacing with the components described above in addition to a SQLite3 database to store waveforms or fingerprints, depending on the method in use (see Section IV-C).

This process differs depending on whether the transmitters may be unknown (*e.g.*, ADS-B) or are all known (*e.g.*, RS-485) and whether there is a claimed identity to be verified (ADS-B) or the harder case in which no identity is claimed.

1) ADS-B: Upon receiving a captured waveform (appropriately masked and with associated metadata), the fingerprinter looks up the claimed ICAO24 address (masked in the waveform but provided in metadata) in the database to determine whether the aircraft has been seen before.

If it has been seen before, there will be a waveform or fingerprint stored in the database. This is retrieved and compared to the incoming waveform; if the similarity score is above a threshold $\alpha = 0.5$ the message is marked as `VERIFIED`, otherwise it is marked as `UNVERIFIED`. If the aircraft has not been seen before, the fingerprinter begins to store waveforms/fingerprints associated with that ICAO24. In the meantime the message is marked as `NEW`: not able to be verified, but not suspicious.

The precise approach taken depends on the choice of fingerprinting method, as described in Section IV-C. In the case of the FIRST method, the first waveform is stored and can then be used for comparison. When using the BEST-OF-N or CENTROID methods, the first N waveforms are stored before the fingerprint is computed. Until then, each incoming message from this aircraft is marked as `NEW`.

All messages receive a verification marking of `NEW`, `VERIFIED` or `UNVERIFIED` as a metadata value. These are made available to any further components relying on them.

2) RS-485: For RS-485, fingerprints should already be known for all transmitters, but for generality no identifier is given in the message to claim which transmitter sent it. Instead, the system performs a comparison with the stored fingerprint of each known transmitter and computes a similarity score. It then provides these scores to downstream components as a list of tuples *(ID, Score)* for each known transmitter.

This enables a trivial determination of the most likely actual transmitter (*i.e.*, that with the highest score) and the confidence in this assessment (*i.e.*, the value of that score).

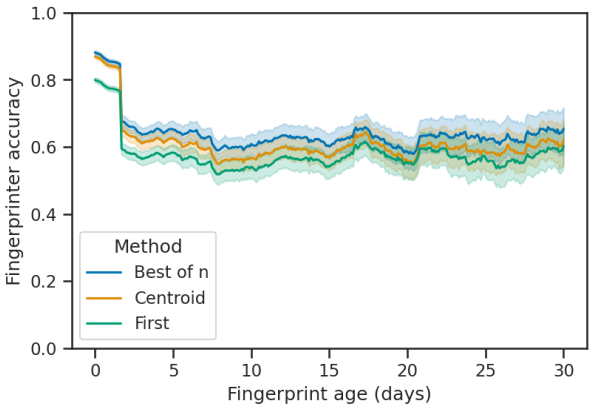

Fig. 7. Comparison of fingerprint age against accuracy of the model for each fingerprinting technique. Accuracy is averaged over 3 days. The shaded region represents the 95th percentile.

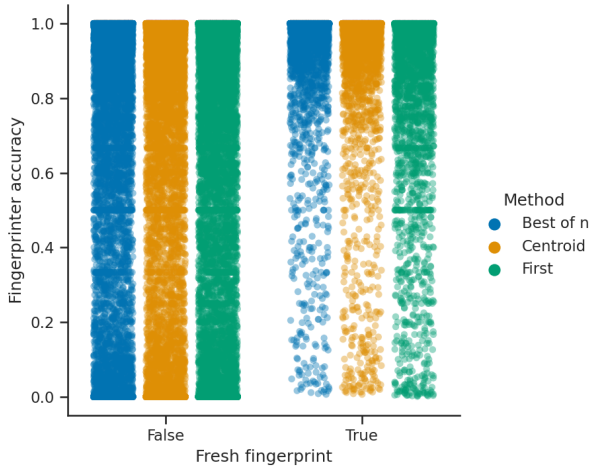

Fig. 8. Strip plot illustrating the distribution of fingerprinter accuracy, comparing flights which are measuring fresh fingerprints against those using historical fingerprint data. There is a clear skew towards higher accuracy when fresh fingerprints are used.

E. Adaptation Effort for New Technology

The system was first implemented for ADS-B and then adapted for RS-485. Not only is this a substantially different technology (wired vs. wireless) but with a different security model as well (known set of transmitters, no given identifiers). Nevertheless, the challenges were relatively minor. In preprocessing: different capture software was used but only as it controlled an oscilloscope instead of an SDR, the demodulator component was replaced with one for the new technology and the filtering and masking components were removed entirely. The only change to the comparison model was to change the input size (from 2400 to 8450) and retrain. The fingerprinter could have remained identical if an RS-485 message protocol had been selected that included transmitter identifiers, but this was omitted to demonstrate the generality of the system and its performance on the harder problem of identification. In practical terms, these tasks took one engineer approximately a week to complete and similar effort would be expected for any other comparable protocol.

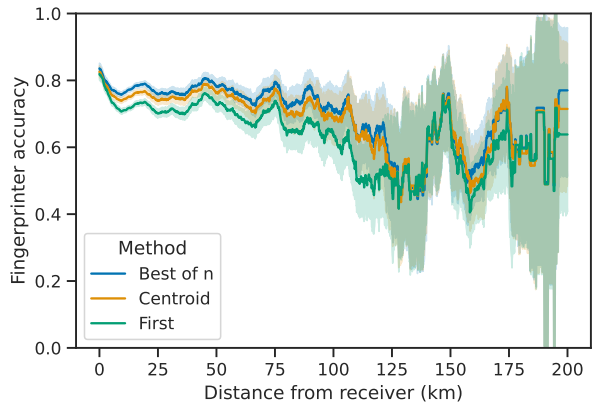

Fig. 9. Comparison of accuracy of the model against distance from the receiver, for each fingerprinting technique. Accuracy is averaged over 10 km intervals. The shaded region represents the 95th percentile.

VI. Evaluation

A. ADS-B

1) Data collection: We performed data collection using the overall pipeline described in Section IV-A, first collecting an initial dataset to train the Siamese model, then subsequently collecting a much larger dataset to test its performance.

The training dataset was collected over 6 days, and consisted of $99,061$ messages from 724 aircraft. The testing dataset was collected over a total of 111 days, within a period spanning 364 days—the dataset was composed of two parts with a significant time gap of over eight months between them, to enable testing of time-based effects on the fingerprint accuracy. Altogether, the testing dataset contained $3,491,116$ messages, forming $71,127$ *flights*—that is, contiguous sequences of messages from the same aircraft within a small timespan. We observed $4,890$ unique aircraft, with each aircraft observed over an average of 14.5 flights and 713.9 messages. Messages were received from as far as 422 km from the receiver, but with 90% of observed aircraft appearing within 54 km of the receiver.

2) Model training: Model training was undertaken using a dedicated computer (*Intel Xeon Silver 4210, NVIDIA Tesla V100*). Training was performed over 50 epochs and completed within approximately 1.5 hours. Before training, data is preprocessed to create sample pairs for the Siamese network. Using training batches of 100 messages, pairs are created such that 50 messages in each batch are paired with messages from the same ICAO24 address, and 50 are paired with messages from a different aircraft.

3) Results: Full data on the accuracy of each method is given in Table II. The best accuracy on the full dataset was 70.2%, produced by an instance using the Best-of-N fingerprint generation approach. This approach produced higher accuracy than the others in the short term (*i.e.*, on the "early" testing dataset) with a peak accuracy of 81.9%, and

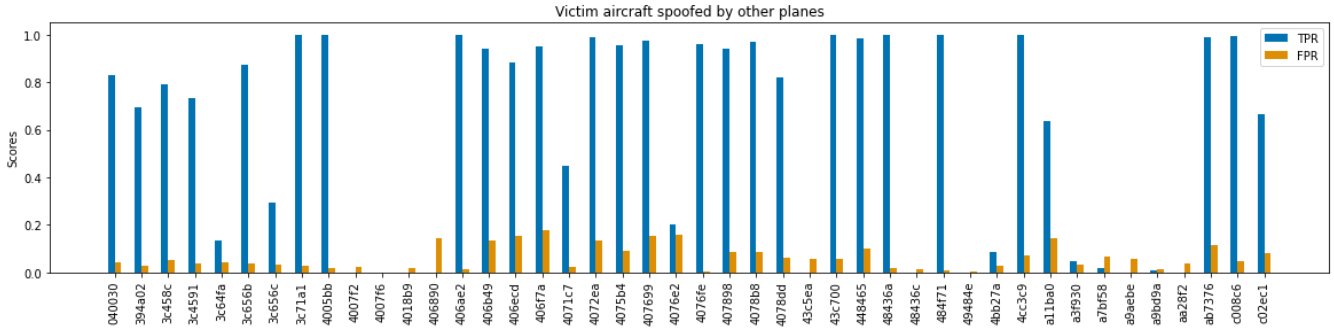

Fig. 10. Adversarial spoofing results. The victim aircraft's identifier is spoofed by each other aircraft. The true positive rate (TPR) shows how well the system is able to verify each legitimate message from the victim. The false positive rate (FPR) demonstrates the attack success rate of adversarial aircraft spoofing the victim's identifier.

sustained in the long term better than the other approaches, reducing to 68.7% when looking at long-term accuracy only.

There are a number of factors which have the potential to affect the accuracy of our model—due to the long duration of our analysis we can see which of these are significant.

a) Fingerprint age: For all fingerprinting methods assessed, the fingerprint is taken from the first few messages received by the aircraft. It therefore seems likely that the time since the fingerprint was captured would affect the accuracy of the model. We observe in Figure 7 that this is not the case—the model's accuracy remains consistent as the fingerprint age increases. This supports our hypothesis that fingerprints are a representation of properties of the transmitter hardware, as they do not appear to change over time.

b) Fresh fingerprints: The accuracy for very new fingerprints is higher, suggesting some amount of the fingerprint is derived from more volatile sources which are matched in the short term but not over a longer period of time. This is also illustrated in Figure 8, which shows that the model is more accurate on flights for which the fingerprint has been taken from that flight. This is an unsurprising result but good to bear in mind—aircraft which have never been seen before should not necessarily be taken into account when assessing the accuracy of the model.

c) Distance from receiver: We also consider the effect on accuracy of the aircraft's distance from our receiver. Illustrated in Figure 9, we see that accuracy decreases slightly as the aircraft gets further from the receiver, and overall variance increases. This can be explained by the fact that signals from further away have to contend with increased background noise, multipath interference, and other factors degrading the signal, making extracting a useful fingerprint more challenging.

d) Adversarial attack: Finally, we consider an attack, in which an adversary flying their aircraft through monitored airspace needs to hide their identity. This is achieved by spoofing the ICAO24 address of their ADS-B transponder. We evaluate our system's performance in this scenario by analysing how well attackers can spoof a selection of 44 randomly chosen victim aircraft. In each adversarial test run, the ICAO24 of each non-victim message is replaced with that of the victim—that is, every other aircraft attempts to impersonate the victim. The verification is considered successful if the Siamese network concludes that the attacked messages do not match the stored fingerprint of the victim aircraft.

We can see in Figure 10 that the adversary's success rate is low throughout, with no victim attacked with a success rate above 19%. Furthermore, the majority of victim aircraft (25 out of 44) are themselves verified, with true positive rates higher than 70%. However, some of the victim aircraft cannot be successfully verified. This is likely due to the impact of performance-degrading conditions discussed above (*e.g.*, a large distance to the receiver), leading to noisy fingerprints that are unhelpful for both legitimate aircraft and attackers.

B. RS-485

We tested our implementation of the RS-485 fingerprinting system in a lab testbed (*cf.*, Section V-B). We used seven serial devices as transmitters communicating on the bus. Four of these devices were of the same model, the other three were from different manufacturers. For each experiment described in this section we captured 1,000 messages per transmitter for training, and the same amount again for testing purposes.

For the noise-free, non-adversarial setting our system achieves perfect fingerprinting performance. We therefore introduced a noise-source to the bus, and tested our system when faced with different noise levels. We tested five noise levels ranging from 0% noise (normal line noise) to 100% noise (the highest amount of noise before decoding errors occur).

In Figure 11, one can see that the noise level affects the performance of the system, with higher noise levels leading to worse performance. We can also see that there is a higher impact when requiring that all bytes of the message have to be verified, going down to 64% for a full message-level verification at 100% noise.

We also notice that identical transmitter units are more affected by higher levels of noise. In Figure 12, one can see that three of the transmitters of the same model (indicated in blue) exhibit a lower successful fingerprinting performance, going as low as 10% for the worst affected transmitter. This is because their fingerprints are more similar as they share the

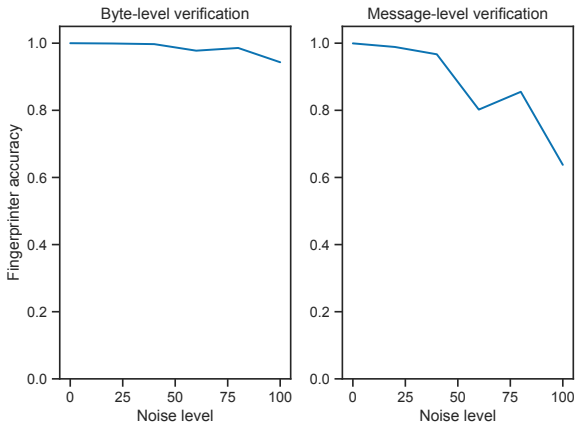

Fig. 11. Noise level affects fingerprinter accuracy. Here, *byte-level verification* means that every byte is verified individually, whereas *message-level verification* means that every byte has to be correctly verified for successful verification. The effect of noise is more pronounced when all bytes of a message have to be successfully verified to pass message-level verification.

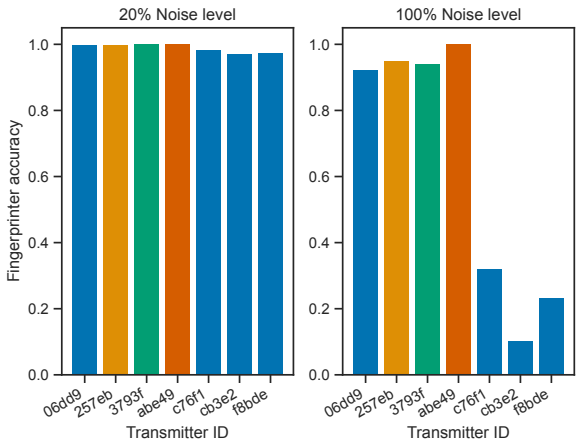

Fig. 12. Comparison showing how different transmitters are affected by the noise level. Transmitters of the same model (indicated by colour) are more affected by noise.

same transmitter hardware design. This lower performance in the presence of noise can be improved drastically by training the system on data collected in a noisy environment. We trained the system on data collected at a noise level of 20%. This improved the fingerprinting performance for all noise levels, with no transmitter having detection rates below 98%.

For our adversarial analysis we considered three types of attackers: two different types of weak attackers, and one strong attacker. The two weak attackers are a compromised device, as well as a device that has been newly connected to the bus by the attacker. In the case of the compromised device, the system has been trained on IQ samples by the compromised device. In contrast, the system has never seen the IQ samples of the newly introduced device. We achieved this by training seven versions of the fingerprinting model, each time removing the data of one of the transmitters. Finally, the strong attacker was implemented by replaying data using an arbitrary waveform generator straight into the bus monitor. This is akin to a strong attacker that attempts to masquerade as a different transmitter by closely mimicking their physical transmission properties. For this analysis we tuned the system to choose the number of verified bytes per message that maximised fingerprinting performance for each noise level. However, while this can improve fingerprinting performance in the presence of noise, this has to be carefully considered as it might still allow more fine-grained attacks to go through.

For attacks using a compromised device, we focused on a noise level of 100%, as for lower noise levels no attack was successful. Even then, only one transmitter had any significant success with an attack success rate of 15% when impersonating a specific transmitter of the same model. Furthermore, when the system has been trained on noisy data, the attack success rate goes down to 0% for all transmitters.

When the attacker can connect a new device to the bus, they have only very low attack success rates when using a different model than the victim device. However, when the attacker uses the same model as the victim device they have a chance to achieve high attack success rates over 90% for exactly one of the legitimate transmitters. This can be mitigated by including a larger number of devices of similar models in the training data. Additionally, intrusion detection systems that detect newly connected devices based on voltage levels such as the system proposed in [4], can be deployed as an orthogonal layer of defence to further mitigate this type of attack.

Finally, we show how a strong attacker can achieve high success rates for all transmitters by using an arbitrary waveform generator (Rigol DG4102) to mimic the physical behaviour of their victim. Figure 13 shows—for a noise level of 20%—that the attack success rate depends on the strictness of the message-level verification (the number of bytes that have to be successfully verified to pass verification). Higher attack precision can be achieved with more sophisticated hardware. This type of attack can be hard to defend against with fingerprinting, but stricter message-level verification can reduce attack success rates. To further improve fingerprinting performance against this type of attacker, the receiver sampling rate needs to be higher than that of the attacker.

VII. Conclusion

In this paper, we introduced a novel hardware fingerprinting approach that can be easily adapted to both radio data links and data bus systems, and thus provides a promising solution to the need to introduce additional security to various legacy aviation systems. We have demonstrated a system that can, with no alterations to the existing hardware and only off-the-shelf equipment, provide additional information on the likely authenticity of messages. We have done so under challenging real-world conditions and over a long period.

In the case of ADS-B, we have observed performance that was adequate under benign conditions, but still had substantial scope for improvement. In adversarial testing, the system

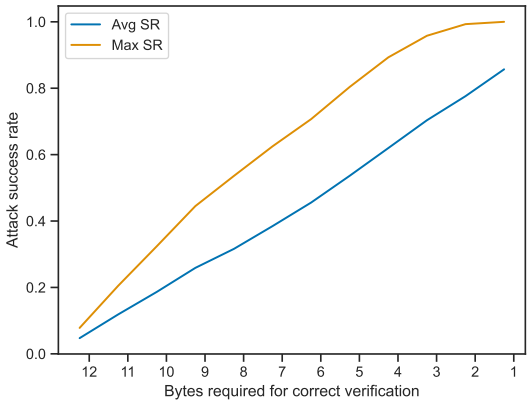

Fig. 13. Average and maximal attack success rates for an attacker using an arbitrary waveform generator at a noise level of 20%. Stricter message-level verification can reduce attack success rates.

performs well, despite the accuracy challenges seen in benign cases. We believe there remains considerable scope to increase the accuracy of the system and that such improvements would assist both the benign-case and malicious-case performance.

For the more constrained setting of an RS-485 bus, we have shown that our system achieves high fingerprinting performance under benign as well as adversarial conditions—even when operating in a very noisy environment. While challenges remain when faced with strong adversaries that can introduce their own hardware, our system nevertheless significantly raises the bar for an attacker.

Based on these two case studies, we show that there is great potential for our proposed system to be useful for the verification of transmissions in a variety of aviation systems. We make our fingerprinting system and the platform that we designed available as open-source to enable further research on our system, or as the basis for applications on other technologies. We are confident that the techniques proposed in this paper can be extended to other radio data link and bus systems. In fact, we have already successfully demonstrated a similar Siamese network based fingerprinting approach in a related context—fingerprinting satellite transmitters [9]. This adaptability had been an initial design goal and we are hopeful that others will also benefit from the platform.

VIII. Acknowledgements

This work was supported by the HICLASS project (113213) funded by Innovate UK.

References

[1] Nimrod Gilboa-Markevich and Avishai Wool. Hardware Fingerprinting for the ARINC 429 Avionic Bus. In *ESORICS 2020*. Springer International Publishing, 2020.

[2] Soorya Gopalakrishnan, Metehan Cekic, and Upamanyu Madhow. Robust wireless fingerprinting via complex-valued neural networks. In *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019.

[3] Mauro Leonardi and Fabrizio Gerardi. Aircraft mode s transponder fingerprinting for intrusion detection. *Aerospace*, 7(3):30, 2020.

[4] Efrat Levy, Nadav Maman, Asaf Shabtai, and Yuval Elovici. AnoMili: Spoofing Prevention and Explainable Anomaly Detection for the 1553 Military Avionic Bus. *CoRR*, abs/2202.06870, 2022.

[5] Karim Lounis, Ziad Mansour, Michael Wrana, Marwa A Elsayed, Steven HH Ding, and Mohammad Zulkernine. A review and analysis of attack vectors on mil-std-1553 communication bus. *IEEE Transactions on Aerospace and Electronic Systems*, 58(6):5586–5606, 2022.

[6] Alessandro Nicolussi, Simon Tanner, and Roger Wattenhofer. Aircraft fingerprinting using deep learning. In *2020 28th European Signal Processing Conference (EUSIPCO)*, pages 740–744. IEEE, 2021.

[7] Francis Onodueze and Darsana Josyula. Anomaly detection on mil-std-1553 dataset using machine learning algorithms. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 592–598. IEEE, 2020.

[8] Harshad Sathaye, Guevara Noubir, and Aanjhan Ranganathan. On the Implications of Spoofing and Jamming Aviation Datalink Applications. In *Proceedings of the 38th Annual Computer Security Applications Conference*, pages 548–560, 2022.

[9] Joshua Smailes, Sebastian Köhler, Simon Birnbach, Martin Strohmeier, and Ivan Martinovic. Watch this space: Securing satellite communication through resilient transmitter fingerprinting. *CoRR*, abs/2305.06947, 2023.

[10] Joshua Smailes, Daniel Moser, Matthew Smith, Martin Strohmeier, Vincent Lenders, and Ivan Martinovic. You talkin' to me? Exploring Practical Attacks on Controller Pilot Data Link Communications. *Proceedings of the 7th ACM on Cyber-Physical System Security Workshop*, 2021.

[11] Orly Stan, Adi Cohen, Yuval Elovici, and Asaf Shabtai. Intrusion detection system for the mil-std-1553 communication bus. *IEEE Transactions on Aerospace and Electronic Systems*, 56(4):3010–3027, 2019.

[12] Martin Strohmeier, Vincent Lenders, and Ivan Martinovic. Security of ADS-B: State of the Art and Beyond. DCS, 2013.

[13] Nagaraja Thanthry and Ravi Pendse. Aviation data networks: security issues and network architecture. In *38th Annual 2004 International Carnahan Conference on Security Technology, 2004.*, pages 77–81. IEEE, 2004.

[14] Xuhang Ying, Joanna Mazer, Giuseppe Bernieri, Mauro Conti, Linda Bushnell, and Radha Poovendran. Detecting ADS-B Spoofing Attacks using Deep Neural Networks. In *2019 IEEE conference on communications and network security (CNS)*, pages 187–195. IEEE, 2019.

[15] Haoran Zha, Qiao Tian, and Yun Lin. Real-world ads-b signal recognition based on radio frequency fingerprinting. In *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, pages 1–6. IEEE, 2020.