Physical layer identification: methodology, security, and origin of variation

by

Ryan Michael Kepke Gerdes

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Electrical Engineering

Program of Study Committee:

Mani Mina, Co-major Professor

Thomas E. Daniels, Co-major Professor

John P. Basart

Dianne H. Cook

Arun Somani

Iowa State University

Ames, Iowa

2011

Copyright © Ryan Michael Kepke Gerdes, 2011. All rights reserved.

DEDICATION

To the Memory of My Father

So, they are not underground,
But as nerves and veins abound
In the growths of upper air,
And they feel the sun and rain,
And the energy again
That made them what they were!

from *Transformations* by Thomas Hardy

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

This dissertation would be greatly diminished if not for the influence the following people have had on it and me.

I thank my co-major professor Dr. Mani Mina for his blend of seriousness and humour across many subjects; without this my graduate studies would not have endured and my views would be even more vulgar, boorish, and parochial. My other co-major Professor, Dr. Thomas E. Daniels, was and is a veritable fount of security knowledge; in too many instances to count, it was he who shaped my nebulous ideas into something concrete through his insistence on rigour, clarity, and simplification.

Each of my committee members helped shape not only my work but also my way of thinking: Dr. Dianne H. Cook showed me the importance of correct methodology; Dr. John P. Basart reminded me that I should take nothing for granted in exposition; and Dr. Arun Somani made available to me his vast professional knowledge.

I also thank Drs. Steve F. Russell and Julie A. Dickerson, as members of my preliminary committee, for their critical and helpful review of my existing and proposed work. Dr. Russell provided the impetus for much of my early work and his interest in its extension and my professional development has always been appreciated.

Several persons provided significant contributions, suggestions, and/or review of individual aspects of this work: Dr. Robert J. Weber of Iowa State University (Chapter 7); Dr. Todd K. Moon of Utah State University (Chapter 6); and Dr. Sasha Kemmet of Iowa State University (Chapter 5).

I express my gratitude, and am indebted, to Jason Sytsma and James Davis for their persistence in collecting data from and on the Ethernet cards used in this study.

My families in the U.S. and Thailand were a reminder to me of what this work is ultimately meant to achieve. To my mother, who has endured and dreamed so much for me, I offer special

thanks. Lastly, my wife remains, quite simply, the best thing in my life.

ABSTRACT

It is common practice to limit solutions for most problems in computer and network security to the purview of the digital domain. Certainly, digital solutions offer much in the way of addressing the security concerns associated with computer and network monitoring and access control. In many areas, however, the available techniques are limited because of their digital nature: authentication schemes are vulnerable to the theft of digital tokens; intrusion detection systems can be thwarted by spoofing or impersonating devices; forensic analysis is incapable of demonstrably tying a particular device to a specific network connection after the fact; and assurance monitoring systems can only provide notification of failures rather than impending failures. In order to address these concerns researchers have proposed, in work falling under the general heading of physical layer identification (PLI), that the signaling behavior of digital devices manifested at the physical layer be used for identification and monitoring purposes.

This work presents a secure methodology, capable of reliably identifying and tracking wired Ethernet cards of the same make and model to a high degree of accuracy, which may be used to corroborate higher layer mechanisms used in authentication and intrusion detection. A framework is also devised, and applied to this methodology, to judge the security of a PLI scheme by determining how resistant it is to forgery attacks using arbitrary waveform generators.

While a PLI scheme must be resistant to attack, it must also be able to identify the preponderance of devices within a given population to be of any practical value. Therefore, a technique, based upon the signalling behaviour specified for Ethernet devices in the IEEE 802.3 standard, is set forth for estimating the theoretical number of devices the methodology is capable of distinguishing between.

Finally, if it can be understood how the individual components of a device give rise to differences in device behaviour, it should be possible to not only model devices for the purposes of

creating new identification and tracking methodologies, but perhaps also allow the community to determine exactly which components should be modified to create device signals resistant to forgery. With this in mind, a methodology is proposed that describes how a device component should be measured to create a model that captures its signalling behaviour, as well as how it can be determined whether or not, and to what extent, the component shapes the device's identity.

CHAPTER 1. OVERVIEW

It is common practice to limit solutions for network-based authentication, and indeed most problems in computer and network security, to the purview of the digital domain. While it is true that working with the digital rather than physical representation of information is perhaps a more tractable approach—one that can, in certain instances, provide definite proof of the soundness of the method or, at the very least, allow us to state that given our current knowledge of various algorithms, theorems, and state-of-the-art technology, our data should be sufficiently well-protected for the next n-years—this emphasis nevertheless overlooks the potential of the physical layer. It instead treats the physical layer as a source of problems or, at best, as something that must be abstracted away.

In the hope of ameliorating such a natural and understandable predilection, this work focuses on differentiating wired Ethernet devices by identifying differences in analogue signalling behaviour, with the goal of using the unique characteristics of signals at the physical layer to corroborate higher layer mechanisms used for authentication, intrusion detection, and assurance monitoring. Specifically, we show that it is possible to discriminate between Ethernet cards of the same make and model and that the signal variability which makes this possible is manifested over a range of spectral and temporal locales.

To set forth but one example where physical layer corroboration could prove useful, consider the limitations of current network access control methods that rely on the use of digital tokens (network keys) or identifiers (usernames and passwords) to prevent unauthorized access. Even strong tokens and identifiers (secure certificates, for example), by their purely digital nature, can be discretely copied if improperly secured and put to use by malicious users. Even worse, popular weak identifiers, such as MAC addresses, can be easily obtained through passive network monitoring and spoofed through the use of a programmable network card. In

contrast, using the unique physical characteristics of devices for identification, in conjunction with current digital practices, could enhance security, as physical features are nearly impossible to obtain (a measurement of the device cannot be done without physical access to the medium) and duplicate.

1.1 Physical layer identification

Building on a tradition dating back to World War II, and inspired by the work of RF engineers in 1990s, researchers of differing fields and expertise began to experiment with identifying network, sensor, and RFID devices in the early to mid 2000s using only the electrical emanations of said devices (see Chapter 2 for a detailed review of this history). As they are primarily concerned with the physical layer of the OSI model, these approaches to identity verification can easily be thought of as *physical-layer identification schemes* (PLIS). The methodology and goals of such works are essentially that of biometrics—identifying devices by non-digital means, though usually only as an aid to existing authentication procedures (by common consensus, they should not be used exclusively for authentication)—with the distinction that the systems under study are non-biological or inanimate.

Physical layer identification, as a technique for intrusion detection, authentication, or forensics, depends upon variability in device behaviour, classifier sensitivity, and the inability of an attacker to adequately emulate device behaviour (for assurance monitoring only the first two are of consequence). Before the security of a particular approach can be evaluated, it is necessary to determine whether devices vary significantly and consistently enough across a given population for a particular identification methodology to reliably differentiate them. Having established this, we must then consider whether device behaviour can be readily emulated. It must be emphasised that the self-evident claim that no two devices can be made to produce exactly the same networking signals is not in itself enough to prove the validity of physical layer identification—signal variation is itself defined with respect to the discriminatory methodology used in identifying devices, while its relevance is determined by the ability or inability of an attacker to produce high fidelity forgeries.

There are three aspects of device variability that need to be considered in relation to physical

layer identification schemes, namely its extent, consistency, and origin. Whether or not we are able to profile and track a device depends upon the device exhibiting its unique behaviour in a consistent manner. To account for the observable amount of change a device undergoes during the course of operation, we must capture the velocity and degree of change during steady state operation and after the device has lost and re-established connectivity. While understanding the origin of such variations may be of use in modelling device behaviour, and hence aid in the construction of device profiles and tracking regimes, it also bears directly upon the security of PLIS. Consider that if an attacker understands the degree to which individual components contribute to differences in device behaviour, they could attempt to engineer devices that conform to known devices by targeting just those components most responsible for expressing identifiable characteristics. Of course, the job of imitating a particular device is considerably easier if the extent of observed variation of a population of devices is not appreciable enough to be discerned by the identification methodology. Determining whether a given methodology is even practicable requires that we test it via application to a statistically significant sample of devices and calculate the theoretical number of devices it is capable of distinguishing between. Such a theoretical analysis is dependent upon the sensitivity of the underlying classification technique(s). The term sensitivity is meant in this context to denote *population* sensitivity, or the number of devices that can be distinguished using a particular methodology.

While establishing that an identification methodology is capable of distinguishing between, and tracking, a population of devices is necessary to prove the security of a PLIS, it is not sufficient. To establish sufficiency it must be determined, according to some measure, how different two signals need be before they can be identified as such. Again, this calculation is dependent upon the sensitivity of the classifier, though in this case sensitivity refers to how close two signals must be in order to produce identical results (we will use the term *difference* sensitivity to avoid ambiguity with population sensitivity). Once this is decided, we must resolve whether an attacker, using current technology, is able produce an acceptable forgery using either an arbitrary waveform generator or by modifying an existing device.

In what follows we investigate the use of PLIS for authentication and forensics (Chapter 4) and propose solutions to address the issues of difference sensitivity (Chapter 5), population

sensitivity (Chapter 6), and origin of variation (Chapter 7).

1.2 Rationale and scope of work

While this work is primarily concerned with the feasibility of identifying Ethernet cards by highlighting differences at the physical layer, our work more broadly attempts to utilise, through the examination of the analogue and digital characteristics of digital devices, the principle of *security through physicality* for such security purposes as intrusion detection (discovering node impersonation and network tampering), authentication (preventing unauthorized access to the physical network), forensic data collection (tying a physical device to a specific network incident), and assurance monitoring (determining whether a device will or is in the process of failing).

Certainly, digital solutions offer much in the way of addressing the security concerns associated with computer and network monitoring and access control, but in each of the aforementioned areas the available techniques are lacking in some respect. We have already mentioned that authentication schemes are vulnerable to the theft of digital tokens (which is not to imply that the physical devices we seek to monitor are immune to theft, only that their absence is more likely to be noticed sooner). Similarly, in other areas of security shortcomings arise because of the limitations of the digital domain: determining the presence of spoofed or impersonated devices for intrusion detection, demonstrably tying a particular device to a specific network connection after the fact in forensic analysis, and addressing impending failures rather than notification of failure in assurance monitoring. In order to address these concerns, we propose making use of the hardware and manufacturing inconsistencies that cause minute and unique variations in the signalling behaviour of every digital device for identification and monitoring purposes. In this respect, our work embodies a certain set of signal processing techniques that seek to make these variations manifest through appropriate application and interpretation.

Central to the security of this method is the belief that these slight variations are difficult, if not impossible, to control and duplicate. This assumption is founded upon knowledge of the variable tolerances of device components, which are introduced in the design and fabrication processes, used in the construction of digital devices. These tolerances allow for unpredictable

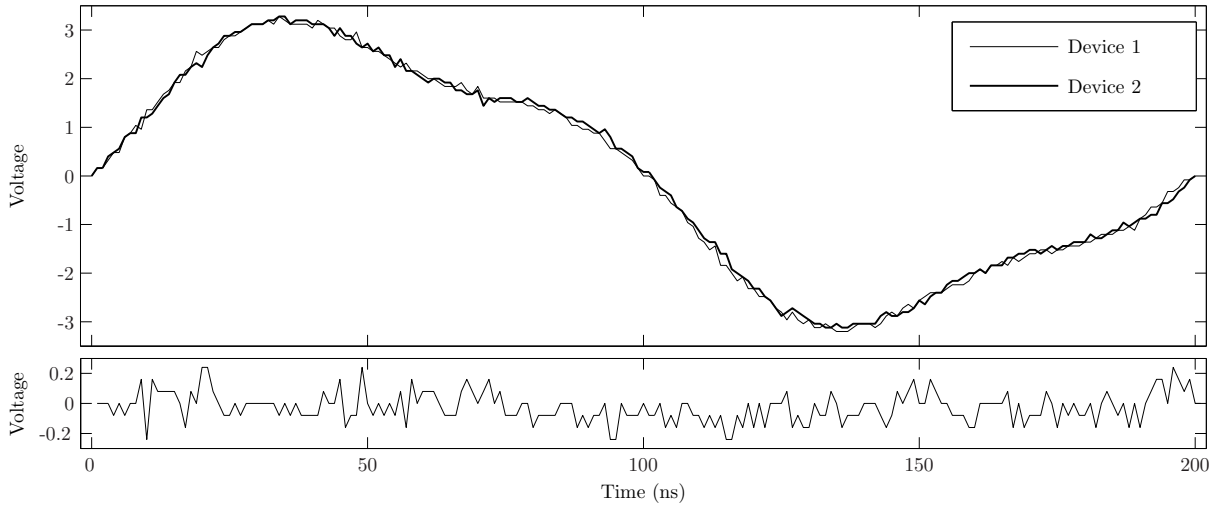

Figure 1.1 (Top) A single period of the synchronisation signals from two 10Mb Ethernet devices, aligned. (Bottom) The difference of the two signals.

variations in the overall electrical operation of the device. Simply put, because of these variations no two devices may be made exactly the same—hence their analogue signal characteristics cannot be made the same, without substantial reverse-engineering beyond the reach of all but the most determined attackers.

Figure 1.1 illustrates the amount of inherent variation (of the time domain) in the signalling behaviour of 10Mb Ethernet devices. The two most important questions concerning signal variation are its extent and consistency; i.e., how appreciable is this variation and is it stable over time? This work is an attempt to answer these questions for the case of 10Mb-compatible Ethernet devices.

While efforts are under way to apply our methodology directly to 100Mb Ethernet and beyond, we chose to study 10Mb Ethernet and leverage it for use in identifying Ethernet cards initially because of the relative simplicity of the electronic devices and signalling involved and its low speed operation. As the electronics and signalling are less complicated than higher-speed systems, we were able to understand the functioning of the devices and identify common behaviour between devices of different makes, which aided us in hypothesis creation and testing while attempting to identify differences and similarities in signals. In addition, as 1Gb cards are required to, and virtually all 100Mb cards do, support 10Mb operation through autonegotiation,

it is possible to force a higher-rate card into 10Mb-mode for the purposes of authentication and then allow it operate at a higher speed thereafter. So long as the link is not severed, it should not be necessary to continuously monitor 10Mb frames. From a complexity and cost perspective, 10Mb Ethernet is also easier to analyse in hardware than other standards due to its simpler modulation scheme, while capturing accurate samples of 10Mb-style Ethernet frames may also be accomplished using lower resolution, slower, and therefore less expensive, ADCs than would be required for higher-speed operation.

Wired Ethernet was chosen due to the low noise environment inherent in wired systems. Environmental noise adds a stochastic and non-stationary component to the signal that must be minimized as much as possible to obtain consistent measurements. On the other hand, noise characteristics of an individual device, or component from a device, may exhibit distinguishing characteristics.

Finally, we believed that if we should fail in discriminating 10Mb Ethernet signals, we would have little chance of succeeding in the high-speed wired and wireless domains. However, we should also note that in some respects profiling 10Mb Ethernet signals may be viewed as a more difficult problem than that of higher-speed systems: the bandwidth is smaller and there are fewer components per device—hence less opportunity for signal variability due to perturbation by device components.

CHAPTER 2. REVIEW OF LITERATURE

Signal detection, classification, and identification was one of the major challenges in the research and development of radar and wireless communication systems for a greater part of the 20th century. In particular, identification of radar, radios, and various wireless communications became a very important and popular topic around the time of World War II [1]. Indeed, research into radar detection and signature evaluation is ongoing in the military to this day [2, 3].

Transmitter identification has also been of interest to the cellular and avionics industries. Patents have been filed that make use of deviations from the carrier frequency in the transient to identify avionics transmitters [4]. In the realm of cellular technology, the steady-state portion of signals has been investigated for the prevention of cellular telephone cloning [5]. A time domain weighted difference technique, whereby the differences between two signals are raised to an arbitrary power and then summed form the fingerprint, was also proposed for the identification of cellular telephones [6]. Additionally, several elaborate systems for the accurate and efficient of collection of wireless signals for use in transmitter identification schemes have been patented [7, 8].

Broadly speaking, we may divide identification and classification techniques into two categories (though there is some overlap between these two in the modulation-based approaches): those that make use of direct measurements of analogue characteristics and those that note digital differences made manifest through aberrations stemming from physical components. The former category may be further subdivided along the lines of the analysis domain (time or frequency) or the portion of the signal used in the analysis (transient or steady-state).

Transient analysis has been and continues to be the most popular: most early methods for radar identification were based upon it and the preponderance of the work reviewed here con-

tinues to rely on it. In fact, most of the modern, academic study of physical layer identification rests upon the identification of radio transmitters undertaken by rf engineers in the 90s [9–11]. The techniques used to discriminate between signals and the types of signals the techniques are applied to are different but the procedure remains the same: transient detection, feature extraction, and classification.

In the area of radio transmitter identification, differences in the local extrema of wavelet coefficients, obtained through the application of the Wavelet Transform to the transient, have been used [11, 12]. Further, genetic algorithms were employed to determine which wavelet coefficients should be used for this type of transient analysis [10]. In attempting to differentiate same and off-model transmitters using a *Sound Blaster 16* sound card for data acquisition, wavelet coefficients were again employed but with neural networks for classification [13]. Multifractal analysis of transients, again using neural networks for classification, have also been used to successfully classify different model transmitters [14]. Amplitude and phase information from the transient envelope [15] and techniques using the maximal difference in amplitude between the same frequency or range of frequencies of two different signals for classification have also been put forth to deal with the identification of radios[16]. Accurate detection of the transient is an essential component of many physical layer identification schemes to this day, and was addressed in the context of radio transmitters by use of a threshold approach [14] and Bayesian step change detector[17].

Using the physical layer for the identification of modern networking devices began with the work of Hall *et al.* In addition to proposing a new transient detection method based upon phase instead of amplitude, they were able to make use of the fractal approach proposed in [14] with the phase characteristics of the transient to differentiate between Bluetooth devices of the same make and model [18]. They extended their work to 802.11b devices by employing a Kalman filter to classify the wavelet coefficients of the frequency, amplitude, and phase characteristics of the transient [19]. A Bayesian filter was used to determine the final classification of a device after transient data from ten transmissions were obtained.

Following this, we proposed a number of techniques for device identification in the wired domain [20] and elucidated an approach to network security and assurance based upon the

physical layer [21, 22].

In a reprise of their work on Bluetooth devices, Hall *et al.* made use of Hotelling's T^2 statistics for classification [23]. The efficacy of this statistic in varying amounts of noise has been subsequently studied for use in the identification of RFID devices [24]. Ureten and Serinken also extended their earlier work on radio transmitter identification to 802.11b devices by applying principal component analysis (PCA) to the amplitude of the transient in conjunction with a neural network for classification [25].

A slightly modified transient detector and the same Kalman filter as Hall *et al.* employed, but with several more characteristics of the transient (length; variability, number of peaks, difference of normalised mean and variance, max value, and wavelet coefficients of the amplitude), were used to identify Chipcon 1000 sensor nodes [26].

Universal software radio peripherals (USRP) have been used to investigate the efficacy of a steady-state approach, whereby a k-NN discriminatory classifier operates on either all of the bins of the normalised FFT of the signal or some combination thereof [27]. This is the only steady-state approach proposed, thus far, for the wireless domain.

While the identification schemes reviewed so far have focused on the physical layer exclusively, a multi-layer framework for intrusion detection was recently proposed in [28]. Focusing on 802.11b and Bluetooth devices, the maximal overlap discrete wavelet transform was used to extract the transient; geometric unsupervised classification was then used on features extracted from the transient using a sliding window, in the time and frequency domains. Higher layer anomalies were produced using simulated traffic and characterised by a Hurst parameter.

Researchers have also investigated the possibility of using arbitrary rf radiation, instead of the transient or steady-state signals that compose the data stream, to aid in the identification of devices. These approaches include having clients measure the effects of multipath on specially constructed frequency probes (i.e., the wireless channel itself is used for authentication) [29] or subjecting devices (specifically, RFID transmitters) to sweeps or bursts of rf energy and measuring the resulting induced emissions [30]. In the latter work, a modified form of PCA is used for feature extraction and Mahalanobis distance for classification.

The affects of the environment on device fingerprints has also begun to be investigated. In

[31], a wavelet smoothed IEEE 802.3 normal link pulse (NLP) was used to identify devices based upon the mean squared error (MSE). The affects of cable length and temperature variation on the MSE were then analysed. In the wireless realm, the mean and covariance of spectral Fisher-features, derived from a linear-discriminant analysis of the relative distance of adjacent Fourier spectra, from the signal transient were used with Mahalanobis distance-based classification to identify Chipcon 2420 sensor nodes [32]. Tests were then carried out to determine the affects of physical distance, antenna polarisation, and battery level on identifying the nodes

While slightly outside the domain of physical layer identification, it has been also shown that it is possible to separate radio stations based upon the shape of the power spectrum [33] and modulation characteristics of the signal envelope [34].

While the preceding works are essentially agnostic to the digital representation of the data being transmitted, recent works have made use of deviations from ideal modulation characteristics. In [35], support vector machines and k-NN classifiers were employed to identify 802.11b devices based upon the average errors across all symbols of phase, magnitude, and error vector magnitudes along with the I/Q origin, carrier frequency, and SYNC correlation offsets of each symbol. Work was also performed on the Wireless Open-Access Research Platform (WARP) using some of the same features but with maximum likelihood estimation used for the comparison of individual features and weighted voting for overall classification [36].

In the category of techniques that make use digital information only, researchers have been able to remotely fingerprinting devices over the Internet by measuring their clock skew through the TCP *timestamps option* [37], even when anonymisation systems are employed [38]. In addition, passive (via probe requests) and active (via frame exchange) forms of timing analyses can be used to identify 802.11b devices and access points [39–41]. Traditional TCP/IP fingerprinting has also inspired researchers to craft non-standard or malformed frames and use a wireless devices response to them in order to determine its firmware or driver [42].

Work in the development of physical authentication schemes has also led to the creation of a physical token that implements a physical one-way function, which is verified using a statistical hashing algorithm [43]. A physical authentication system was introduced in [44] that identified FPGAs based upon the delay characteristics of the logic path in the integrated circuit used

to compute 128 hash-based challenge-response pairs. In related work, it was suggested that delays be intentionally added to the logic circuitry in order to improve security [45]. Using these techniques it is also possible for devices and another entities to generate mutually known shared secrets [46].

The central premise of PLIS is that reproducing device-specific variation is difficult and not cost effective. Challenging these assumptions are recent works by Danev *et al.* and Edman and Yener [47, 48]. These attacks can be divided by the equipment used to carry them out (universal software radio peripherals [USRP] or arbitrary waveform generators) and also the style of attack (replay of observed signals or creation of new signals based upon the features of the observed signals). In [47, 48] the radiometric PLIS proposed in [35] was compromised by both replay attacks and the creation of new signals; both the replay attack and generation attack were carried out using USRP in [48], while [47] utilised an arbitrary waveform generator (AWG) for the replay attack and USRP for the generation type. Furthermore, in [47] Danev *et al.* also showed that an AWG could be used to successfully attack the transient approach given in [32].

CHAPTER 3. METHODOLOGY

The methodology adopted for the application of *inanimetrics* to a specific type of technology is: (1) identify and acquire a recurring and ubiquitous signal, $\mathcal{S}$, to serve as a 'fingerprint', (2) extract a set of features from the signal, $L = f(\mathcal{S})$, and (3) employ a classification technique to compare a test feature set with a database of existing feature sets in order to verify the purported identity of the test subject. It is this last item that we wish to discuss.

For rather than following the best practices of the biometrics community (making use of threshold or hybrid techniques to compare feature sets [49]) researchers, from all backgrounds, working in the area of PLIS have consistently made use of rank-based classification techniques that are only applicable under a closed-world model. This problem, though endemic, does not invalidate much of the innovative and interesting work done in this field; indeed, rectification of the problems related to verification will probably only result in a slight increase in false-accept and false-reject rates for many works[1]. It does, however, point to the need to re-examine the progress made in the field of PLIS over these past few years and its future direction.

In what follows, we detail the implicit assumption (that of a closed world) made by many researchers through an examination of the techniques used to verify the identity of a subject device. We have chosen—based upon historical importance, novelty, and prominence—what we believe to be a foundational sample of papers to critique. In addition, we provide a list, by no means definitive, of other works affected by our critique. Finally, we present a methodology appropriate for classification in PLIS.

[1] In examining whether or not the radiometric fingerprinting system of Brik et al. [35] is vulnerable to forged frames, Danev et al. implemented a threshold-based scheme and achieved results in support of this assertion (see section 5 of [47]).

3.1 The open- vs. closed-world assumption

The decisive factor in the selection of an appropriate technique for use in identity verification depends upon the assumption of either an open or closed world [49]. These two world models are delineated by who, in the case of biometrics, has access to the authentication system: in the closed world it is assumed only users already enrolled in the system may present a biometric identifier to it for verification, while in an open world anyone (even those not enrolled) may access the system for the same purpose. It follows that in a closed world it is never acceptable for the system not to map an identifier to a valid user, whereas an open-world system must be able to assert that an identifier does not belong to a valid user if the identifier is absent from the database. If PLIS are to be used for device authentication and network access control, they must adopt the open-world assumption.

To illustrate the affects of assuming a closed world for a PLIS, consider a system of n-devices which provides authentication services by verifying that an asserted MAC address maps to the proper device based upon signal characteristics. The system would consist of a feature set derived from each device's signal, $\{L_1, ..., L_n\}$, and a corresponding list of MAC addresses, $\{M_1, ..., M_n\}$. Operating in a closed world, an attacker need only submit frames using each of the MAC addresses $\{M_1, ..., M_n\}$ until a positive response is received. It is necessarily the case that one of the forged MAC addresses will be associated with some device's feature set, as per the definition of the closed world. The entire security of the system then relies merely upon the attacker not knowing the MAC addresses of the devices associated with it.

In biometrics three approaches for comparing feature sets are possible: threshold, rank-based, and hybrid [49]. Of the three, only the threshold and hybrid techniques are appropriate for an open world. Thus, if classifier appropriate for use only in a closed world is used in a PLIS, the closed world has been assumed, de facto. All of the PLIS under consideration here either make use of a rank-based classifiers or hybrid classifiers in a rank-based manner for verification. The efficacy, computational complexity, and memory footprint of a particular classifier are secondary concerns and must, sometimes, be sacrificed to meet the conditions of an open world.

Again, our intent is merely to highlight the fact that in each of the works reviewed a closed-world assumption has been made, and that while this does not invalidate the work as a whole, it nonetheless requires a re-working of the verification method before the schemes can be seen as viable for authentication or intrusion detection purposes.

3.2 Closed-world PLIS

For the purposes of this analysis, the feature set used and the nature of their extraction is inconsequential, though this is not the case in general. If features are selected in such a way that they are wholly dependent upon the underlying data used in classification (say, for instance, a single data point or collection of points of a signal are found via searching to be effective at discriminating a collection of devices) then the conditions of a closed world criteria are met, which would invalidate the feature extraction technique.

For ease of interpretation, we chose to standardise the notation of the works analysed. In what follows, L_i^j is used to represent the feature vector derived from the j^{th} frame of the i^{th} device; $L_i^j(k)$ denotes access to the k^{th} element of the feature vector. In addition, a collection of feature vectors from the i^{th} device are denoted as $\mathbf{L}_i$, where $\mathbf{L}_i(j)$ is used to refer to the individual vector L_i^j. The feature vectors of frames that are to be tested by the system are always accompanied by the subscript T.

Brik et al.

In [35], k-nearest neighbour (k-NN) and support vector machines (SVM) are considered separately for identity verification. In the case of k-NN, the identity of a collection of frames (which is to say that multiple frames are required to verify the identity of the transmitting device) is established by comparing n-test frames, all of the same source MAC address, to m-training frames of every device in the system using the Manhattan distance metric

$$d\left(L_i^j, L_T^k\right) = \sum_l \left| L_i^j(l) - L_T^k(l) \right| \tag{3.1}$$

The test frames are then assigned to the device most frequently associated with the minimum distance between test and training vectors, or in the case of a tie the device with 'the great-

est cumulative similarity'. The procedure is much the same for use of SVM (simply replace Manhattan distance with SVM in the above).

The defect of the approach lies in the fact that the test frames are compared to every device in the system, with the identity chosen as the closet match between feature vectors—no threshold is used to determine when the closest match is still too distant and thus a match will always be made. The authors do mention that for the SVM approach such a threshold could be established but fail to mention how this is to be effected (i.e., how SVM are to be used in a hybrid mode). In fact, the entire apparatus could be simplified, at least for k-NN, if a threshold-based approach were used from the beginning. Interestingly, this very idea is derivable from an ambiguous assertion intended by the authors to prevent forged frames from being considered by the k-NN classifier.

Before discussing the details of how k-NN and SVM can be used for identification at the end of Section 3.4, the authors state that in order

> [t]o guard against impostors injecting frames with forged MAC address [sic] in the [identity verification routine], the server may choose to discard outliers before running classification on the [test frames].

The authors intention, we believe, is that an outlier removal algorithm, used earlier in the training process for k-NN, is to be used on the n-test frames, only some of which are legitimate, to eliminate forged frames before they are used in the identification process. During the training phase, half of the original training data is reduced by discarding each L_i^j that satisfies

$$\arg\max_{L_i^j} d\left(\frac{1}{m}\sum_k L_i^k, L_i^j\right) \tag{3.2}$$

where *d()* is the Manhattan distance and m is decremented at each iteration until $m/2$ frames remain. A similar procedure applied to test frames might possibly work, though it must still be decided how far away from the average frames should be in order to be disregarded, but only if the preponderance of test frames are legitimate—if most frames were forged then only the legitimate ones would be discarded. A far better approach, one that obviates the need for the cumbersome k-NN approach described above, is to compare the test feature vectors to just the

training vectors of their purported identity using Manhattan distance, along with a threshold for maximum allowable distance from the training vectors. In Section 3.3, we provide details on how such a threshold could be established.

Hall et al.

In some of the earliest work on PLIS [19], Hall et al. utilise Kalman and Bayesian filters in tandem for identity verification. An initial guess, P_i^j, that the j^{th} frame originated from the i^{th} device (this procedure is carried out against all of the devices in the system) is computed using a Kalman filter for each of n-test frames. P_i^j is given by the filter

$$P_i^j(L_T^j) = \exp -\frac{1}{2}\left(L_T^j - \widehat{L}_i\right)^t \Sigma_{\mathbf{L}_i}^{-1}\left(L_T^j - \widehat{L}_i\right) \tag{3.3}$$

where $\widehat{L}_i$ is the average of m-training feature vectors, $\mathbf{L}_i$, and $\Sigma_{\mathbf{L}_i}$ their covariance. The individual guesses for each frame and device are then used to determine the identity of the frames by associating the frames with the device that maximises the output of the Bayesian filter

$$B(P_i^j) = \begin{cases} P_i^j B\left(P_i^{j-1}\right) & \text{for} \quad j > 0 \\ \frac{1}{\#\text{ devices}} & \text{for} \quad j = 0 \end{cases} \tag{3.4}$$

for $j = n$.

In the above, the closed-world assumption is made twice: once by calculating probabilities for a single frame across all devices and again in making use of these probabilities to select the maximum output of a Bayesian filter applied to each device.

Other works

We would also like to point out that several other authors also appear to assume a closed world in their work [25–27, 36]. A valid threshold-based approach is given in [24, 30, 32], and while [23, 28] seem to make use of thresholds, more information on their derivation is needed before a final judgement can be made as to their validity.

3.3 Open-world PLIS

In all of the papers cited, an inappropriate classification technique was used for verification; a closed world was implicitly assumed through the use of a rank-based classifier. When the world is opened, which is necessarily the case for authentication and intrusion detection, a rank-based system *unavoidably* allows in impostors. As such, each of the these PLIS must adopt a threshold or hybrid approach for verification.

While hybrid approaches are probably more powerful—sophisticated learning techniques can be brought to bear on the problem of differentiating the feature sets of highly similar devices—they incur substantial a memory penalty (more data needs to be stored: not only training data and thresholds, but also a logical structure that reflects how the underlying data for each device should appear independently and in relation to data from other devices) and increased computational complexity (all test data must be passed through multilayered tests and/or compared to the training data of several, if not all, of the other devices in the system) over straight-forward thresholds. For these reasons, we have chosen to make use of thresholds in our work [20, 31, 50].

In what follows, we present a threshold-based approach for identity verification that is agnostic to the underlying feature set and appropriate as a drop-in replacement of the classification techniques used by many of the above works. A threshold scheme is broadly and easily applicable so long as a distance metric is employed in comparing the feature vectors of test and training frames. In general, the distance metric is also immaterial, so long as it is only used to calculate the distance between a test feature vector and a training vector from a single device (this is the case for the Manhattan distance metric; even the Kalman filter of Hall et al. may be interpreted as a distance metric of sorts, as it returns a probability of nearness).

Details of our approach are given in [50], but the procedure is briefly this: Acquire training data, $\mathbf{L}_i = \{L_i^1 \cdots L_i^m\}$, for a device by extracting the necessary features from several captured frames and use some of them (by choosing a feature vector at random or performing some sort of averaging on a selection of vectors, e.g.) to establish a reference feature vector, L_i^R. The remaining vectors are then used to determine a baseline for the device by calculating the

distances, using an arbitrary distance metric, between the reference vector and training vectors, $b_i = d\left(L_i^R, \mathbf{L}_i(j)\right)$ for $j = 1 \cdots m$. Finally, using the baseline values, calculate a threshold, $th_i = f\left(b_i\right)$, that specifies the maximum allowable difference between the reference and any future test feature vectors. If the distance between the two is greater than the threshold, the frames are to be rejected as not having originated from the device. It should be noted that test frames are only compared to the reference vector of their asserted identity.

Thresholds may be deduced experimentally or derived analytically through modelling of device, signal, or feature behaviours; we have found statistical intervals [51] especially useful in the former case. An important consideration in selecting a thresholding approach is deciding whether a dynamic threshold is needed or a static one will suffice. It is essential that repeated measurement of devices, made over the medium-term, be undertaken to understand if the feature set changes due to outside processes or influences, such as multipath, interference, noise, or temperature variation. It has been our experience that thresholds will need to be updated to adapt to changing device behaviour.

If multiple frames are needed for verification the frames could either be averaged before comparison or a distribution of the differences between test vectors and the reference could be constructed and then compared to a distribution built from the baseline differences of the device using a distributional comparison test, such as Kolmogorov-Smirnov. In the case of multiple features, where distances aren't computed using a vector-based metric, it is possible to construct baselines for each of the features separately and, having compared the individual test features to the appropriate baseline, simply count the number of features that fall within the thresholds. If that number is sufficiently high, as determined by some other threshold, the frame(s) could be accepted as legitimate.

Another possible solution is to employ the threshold determination techniques used in the biometrics community, as exemplified by Danev et al. [30, 32]. Chapter 5 of [49] provides an excellent introduction to this approach, but in brief the idea is to find a threshold such that the false-accept rate of the system equals the false-reject rate. While Danev et al. have demonstrated the viability of such an approach for PLIS, at least in the short term, it has been our experience, having observed devices continuously for several hours, that all devices undergo

constant, subtle changes that affect the fingerprint and therefore necessitate the use of adaptive thresholds if the system is to identify devices over long periods [50]. While it is possible to use thresholds based upon overall system performance to take into account changes of the underlying signal, it is simpler, we feel, to replace them with adaptive per-device thresholds of the kind outlined above.

3.4 Conclusion

We have attempted to review the methods used for identity verification in several physical-layer identification schemes. Though never mentioned explicitly, the incorrect assumption of a closed world was made manifest through the use of inappropriate classifiers in each of the preceding works. Our purpose was not to denigrate these works, much less suggest that their results should discounted; rather, we had hoped that by pointing out and offering a suggestion for correcting this oversight, existing work in the area of PLIS could be preserved and strengthened.

CHAPTER 4. IDENTIFICATION

The proposed approach to device-level authentication makes use of anomaly detection based upon the layer-specific behaviour of the Physical and Data Link layers; our intent is the 1:1 verification of device identity and not the classification of devices. Verification is achieved by monitoring the analogue characteristics of the synchronisation signal used in 10Mb Ethernet systems and linking changes in MAC addresses to their corresponding physical representation to, for example, determine whether the digital hardware address matches the expected physical signal. A profile, representing the expected device behaviour, is generated from training data before identification of possible anomalous activity begins. This profile is used to determine whether new information, in the form of electrical signals received from the device being monitored, conforms to previous behaviour. Periodically the device's profile is updated to compensate for the expected drift in its performance.

In the present work, an optimal detector, the matched filter, is used to create profiles of signals by applying the filter to various sections of a 10Mb Ethernet frame. The output of the filter, as well as the expected deviance from this output, is used in the construction of the profile.

In contradistinction to existing work, we focus on the use of the steady-state portion of signals to differentiate highly similar devices, where the decision to accept or reject the purported identity of a device is based upon an analysis of only a single Ethernet frame. In an attempt to create a worst-case scenario (viz., one in which an attacker is able to acquire a contemporaneously manufactured device), the cards chosen for our study were purchased in bulk and, within each model class, bear nearly identical chipset markings and proximate MAC addresses and serial numbers. We also present an adaptive thresholding scheme—which is readily applicable to other physical-layer identification work—capable of capturing the signal variation seen

during medium-term observation of the devices. While our methodology is comparable to that of biometrics [49], it does deviate from the biometric approach during the evaluation stage so as to accommodate this thresholding scheme. Finally, we carry out several experiments meant to provide insight into which areas of the 10Mb Ethernet signal are useful for identification purposes.

The rest of the chapter is structured as follows: We supply the background material underpinning the work in Section 4.1. Following this, in Section 4.2, we detail the steps necessary to identify a device by its signalling characteristics using our methodology. The experimental approach used to evaluate the efficacy this methodology is supplied in Section 4.3, with the details of the Ethernet cards used and an interpretation of the subsequent results appearing in Section 4.4. Finally, we conclude by noting the open problems and a shortcoming of our work; future avenues of research are proposed to address these.

4.1 Background material: matched filters

Commonly used in receiver systems, the matched filter is said to be an optimal linear detector as it can be shown by the Cauchy-Schwarz inequality that the filter maximizes the signal-to-noise ratio of a known input signal in additive white Gaussian noise (AWGN) [52]. The transfer function of the matched filter, in the frequency domain, may be stated as

$$H(\omega) = \kappa \frac{A^*(\omega)}{P(\omega)} \exp^{-j\omega t_0} \tag{4.1}$$

where $A^*(\omega)$ is the complex conjugate of the Fourier Transform of a known time-domain signal $\alpha(t)$, $P(\omega)$ is the *power spectral density* (PSD) of the noise associated with an input signal, κ is an arbitrary constant, and t_0 is the sampling time of the peak filter output. This latter variable is determined in advance as the filter output is maximised when the received signal is aligned with the known signal, which can be accomplished through the use of synchronised clocks at the transmitter and receiver.

By selecting an appropriate value of κ for the operating environment, and assuming AWGN for the PSD, $P(\omega)$ may be eliminated from (4.1). For a given input signal, $\beta(t)$, the output of the filter, $\gamma(t)$, at sampling time t_0 for the Gaussian noise case, is found using the convolution

operation in the frequency domain [53]

$$\begin{aligned}\gamma(t_0) &= \frac{1}{2\pi}\int_{-\infty}^{+\infty} H(\omega)B(\omega)\exp^{j\omega t_0}\,\mathrm{d}\omega \\ &= \frac{1}{2\pi}\int_{-\infty}^{+\infty} A^*(\omega)B(\omega)\mathrm{d}\omega\end{aligned} \tag{4.2}$$

where $B(\omega)$ is the Fourier Transform of the time-domain input signal $\beta(t)$.

Taking the inverse Fourier Transform of (4.1) gives the transfer function of the filter, $h(t)$, in the time-domain (again for the AWGN case) as

$$h(t) = \begin{cases} \alpha(t_0 - t), & 0 \leq t \leq T \\ 0, & \text{elsewhere} \end{cases} \tag{4.3}$$

where T is the period of the known time-domain signal $\alpha(t)$. Convolving (4.3) with $\beta(t)$ expresses the filter application in the time-domain

$$\gamma(t_0) = h(t_0) \star \beta(t_0) = \int_{t_0-T}^{t_0} \alpha(\tau)\beta(\tau)\mathrm{d}\tau \tag{4.4}$$

Thus, the matched filter may be interpreted as the inner-product (in a discrete time formulation, the dot-product) of two time-aligned signals, or a simple correlation.

It should be noted that we have used, and will continue to use for the sake of consistency, a continuous time formulation in our discussion of systems, even though the implementation of the methods we present occur exclusively within a discrete time framework. As such, when giving interpretation or explanation, we will occasionally lapse into discrete terminology for the sake of clarity.

4.2 The signal profile

We describe how the matched filter may be used to create a signal profile useful for identifying a signal's device of origin.

4.2.1 Identifying a common signal

In order to create a profile of the signal characteristics for an Ethernet device, a portion of the frame common to all devices was identified. In this context, commonality is used to denote

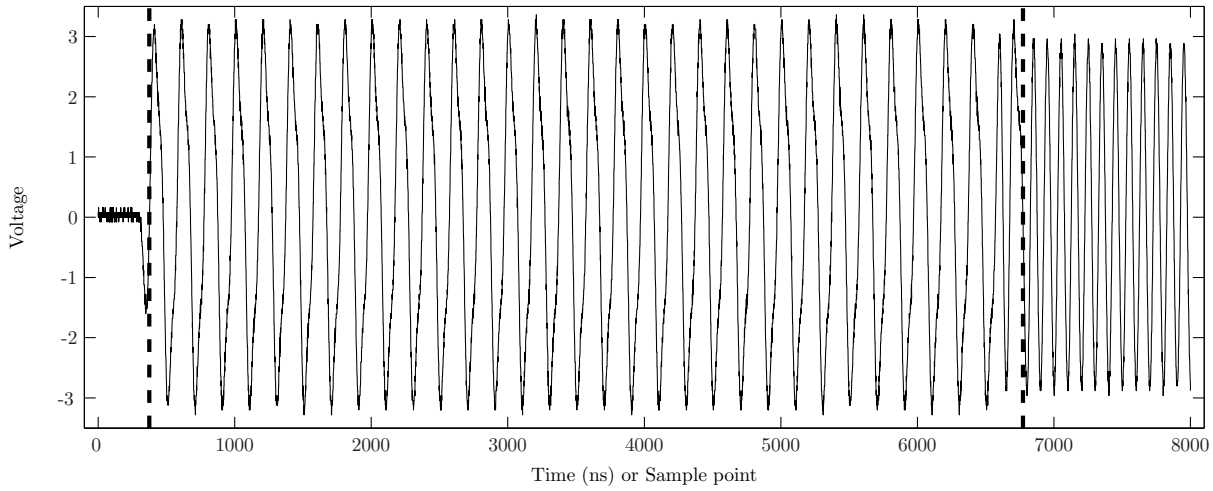

Figure 4.1 Beginning of an Ethernet frame (dataset1): (left) noise and transient, (centre) synchronisation signal, and (right) destination MAC address.

two distinct and necessary requirements: recurrence and ubiquity (i.e., the common portion of the frame must appear at the beginning of each frame for every device). The *preamble* of the frame or, colloquially, the *synchronisation signal* (named as such because it is sent to synchronise the receiver of the destination device to the transmitter of the source device) satisfies both of these requirements. This signal precedes each frame and consists of a 56-bit sequence of alternating ones and zeros, encoded using Manchester encoding with a fundamental frequency of 5 MHz, and is followed by the *start frame sequence*, `10101011`, which is used to indicate that the rest of the frame follows (Figure 4.1 and Figure 4.2). The signal consists of a transient, or turn-on, portion, which is the result of the transmitting circuitry of the sending device powering on, as well as a steady state portion that serves as the actual synchronisation signal.

As mentioned earlier, most work in signal identification has traditionally focused on the transient portion of a signal. However, as the transient signal in 10Mb Ethernet is so small, in terms of the number of wavelengths of the overall signal, we do not believe that there is physically enough information contained in it for the identification of similar devices. As such, our methodology relies primarily upon the steady-state portion of the signal for profiling purposes.

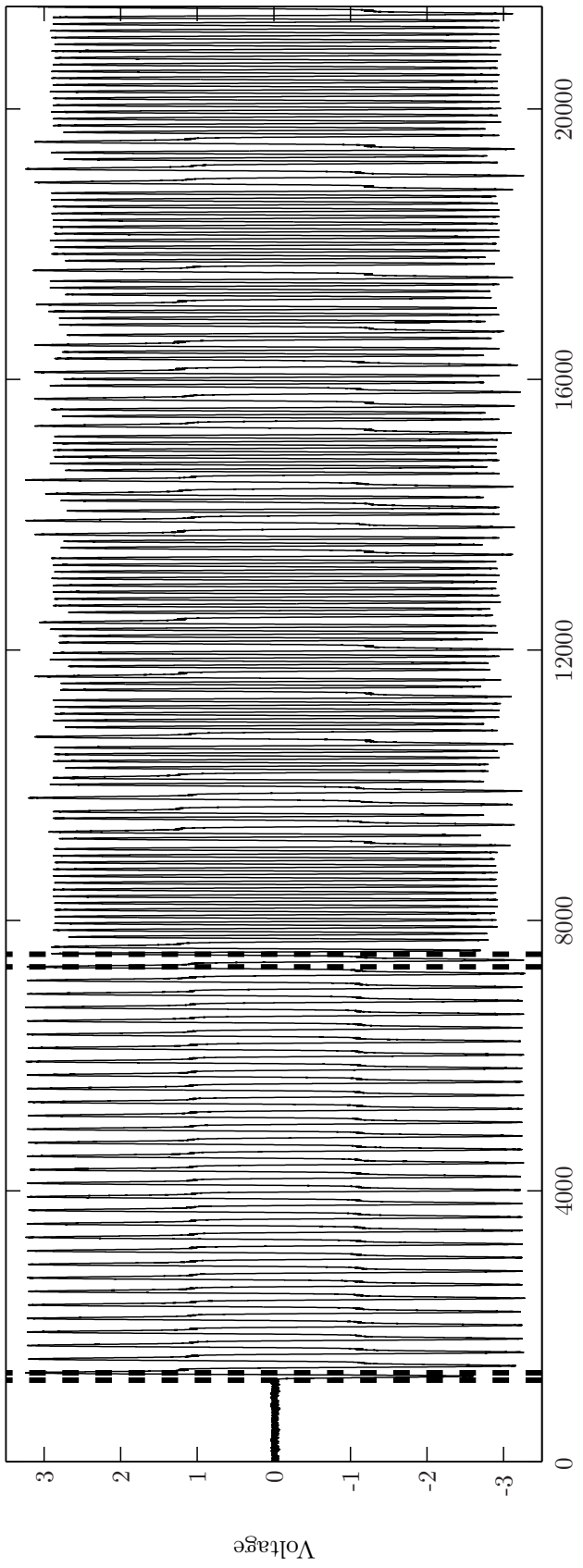

Figure 4.2 Beginning of an Ethernet frame (dataset2/3): (left) noise then transient, (centre) synchronisation signal then transition to MAC addresses, and (right) destination MAC address and portion of source address.

The final portion of the Ethernet frame shown in Figure 4.1 is the beginning of the MAC address of the receiving device.

4.2.2 Creating a signal profile

We discuss the three components of the signal profile and their purposes: the matched filter (the anticipated signal); the filter output (used to characterise a signal); and thresholds for the filter output (needed to identify and track).

4.2.2.1 The matched filter

Having identified a common and repetitive portion of the Ethernet signal suitable for identification purposes, an exact starting position and period of the portion of the signal to be matched to must be chosen. We call this part of the signal the *reference signal* and choose it to represent the known time-domain signal $\alpha(t)$. As per (4.3), the reference signal must be reversed in the time-domain and shifted by t_0 to be used as the filter. Here t_0 is to be regarded as the final sample point or end time of the reference signal.

Initially, the period and position of the reference signal were chosen as an arbitrary number of points spanning the length of the synchronisation signal. For 10Mb Ethernet at least, we have found this acceptable to distinguish between all but the most similar of signals; however, we have also developed algorithms to determine the optimal reference signal for a set of known devices. This type of reference selection would be useful for a network of homogeneous devices (e.g., a lab environment in which all of the computers use identical Ethernet cards and were purchased at the same time) that might be assumed to act as a representative sample, and where, positing a worst-case scenario, an attacker would have access to like devices. For a general study of the matched filter, however, we have selected a reference signal that includes only the steady-state portion of the synchronisation signal, which is the same, to within five sample points (5 ns), for each device. While optimally determining a reference signal for a device, in relation to other known devices, may increase performance, our experiments have shown that it is not generally necessary to do so.

4.2.2.2 Filter output

The next step in creating a signal profile is to apply the filter to the signal used to create it; i.e., convolve the filter with the portion of the signal used for the selection of the reference signal. The filter returns a single value from this operation that serves as a baseline. This value represents the filter response when a perfect match is made between the filter and the original signal. If another signal is exactly the same as the original, we expect that applying the filter to this signal will produce the same value. In general then, applying the filter to a signal produces a measurement of the closeness of the signal to the original and, consequently, the alikeness of the devices the signals were acquired from. If a signal from a different device approaches the filter output value for the original signal too closely then we would be unable to distinguish it from the device that produced the original.

Due to the noise inherent in any system, we cannot assume that even a properly functioning device will output exactly the same synchronisation signal for each frame. Noise from surrounding devices, created by a hard disc or CD-ROM being accessed or variations in system load, and thermal noise assuredly cause slight variations in the signal from frame to frame. In fact, with the aid of temperature recording equipment we have been able to correlate aberrations in the filter output to variations in the ambient temperature of the lab. Furthermore, due to the non-ideal properties of the Ethernet cabling—parasitic resistance, capacitance, and inductance—even the act of measuring the signal on a different portion of the Ethernet cabling, or using a different cable altogether, may affect the measured signal. This affect, however, gives rise to the interesting possibility of detecting passive taps on the line, which often change the effective material parameters of the medium.

To take into account the inherent variability of every device's output, as well as external factors such as temperature and system load variations, the signal profile must include the filter outputs from a collection of signals taken over a period of time (the filter created by the original signal is applied to the collection of signals and the response to each recorded [Figure 4.3]). We have found that only 25 sequentially sampled signals are necessary to adequately characterize the unique signalling behaviour of a device using a matched filter.

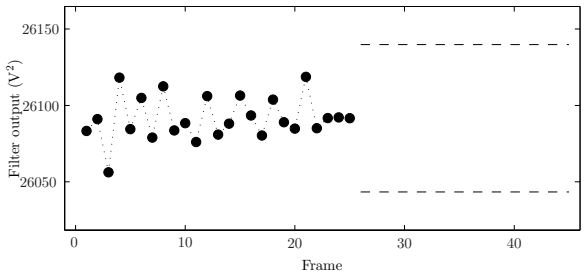

Figure 4.3 Filter output for 25 frames of an Ethernet device (*cve* 0.0011); the outputs for frames 26–45 must lie between the dashed lines.

By examining the filter response for a device over a number of hours, we have determined that a device's synchronisation signal undergoes a continuous, if slight, change. In many cases we have discovered that slight variations in the amplitude of the signals are the cause of this variation. A subtle change in signal shape, over a period of hours, also changes the filter response. By using the average of several synchronisation signals for the reference signal we have been able to decrease the variation of the filter response; however, this often leads to a corresponding increase in false negatives.

4.2.2.3 Thresholds

The final component of the signal profile is the establishment of thresholds that determine the extent to which future outputs of the filter for a device may vary from the baseline. In other words, the maximum amount of deviation in filter response acceptable before a signal is deemed too different from the original must somehow be quantified (the dashed lines of Figure 4.3). In our approach, which is detailed in Section 4.3.3, these thresholds are deduced from the distributional properties of the recorded filter responses mentioned in the previous section. In addition, as the underlying signal used for profiling is stochastic, the thresholds for the allowable range of filter output will indubitably change over time; thus, it is necessary that some method of tracking output and adjusting thresholds accordingly be devised. As such, we require that the next m-frames resemble the previous n-frames, where thresholds are to be updated every n-frames. In this way—by taking into account past behaviour and projecting it into the near-term, with a tolerance added for expected variation—a device can be adaptively

tracked as its signal changes over time.

Moreover, by using thresholds to set bounds on future outputs, it is possible to adjudge the origin of a signal on a frame-by-frame basis. For especially similar devices, however, it may be necessary to forfeit unitary action and make the decision to accept the signal as having arisen from the filter's device of origin, or reject it as too dissimilar, using multiple outputs, with the hope that any slight differences will compound and manifest themselves over time. To this end, a distributional comparison test, such as the two-sample *Kolmogorov–Smirnov* test, could be used to compare the outputs of the filter when applied to frames gathered during an observation period to the filter responses in the profile. If the null hypothesis is not rejected for a given significance level, then the frames would either be transmitted (if they had been queued) or continued connectivity be granted to the device. It must be said, though, that this scheme is somewhat disadvantageous, in that it allows spurious frames through during the observation period; as such, we have chosen to make the decision to accept or reject a signal on a frame-by-frame basis.

4.2.3 Pre-processing data

In an attempt to improve the efficacy of our method, which we shall term the generic matched filter approach for the remainder of this work, we devised several ways to pre-process signals to improve our ability to discriminate between highly similar devices. Each of these techniques seeks to amplify slight differences in signal characteristics that are too subtle to be distinguished by the original method and are usually applied to both the reference and test signals. The impetus for this work was based upon the observation that as the matched filter operation is a sum of products (viewed as such in the discrete case), large-scale similarities between signals can often overshadow the small-scale differences useful for signal profiling.

4.2.3.1 An ensemble of filters

For a given device multiple matched filters may be created by selecting a reference signal for each portion of the signal identified in Section 4.2.1. Matching filters to the transient, steady-state, and source MAC address sections of the frame gives a full characterization of the

broad traits of a signal. An ensemble of filters is utilized, instead of a single large filter, so that strong similarities in certain regions of the signal cannot overshadow smaller differences in others.

Selecting multiple reference signals for each section of the signal may also highlight slight differences; e.g., each transition, or pair of transitions, of the synchronisation signal could be matched to different filters. In such a way the granularity of filtering could be arbitrarily increased to take into account the smallest of differences.

4.2.3.2 Bandpass filtering

By analysing the spectrum of signals from a multitude of similar devices, we have found that distinguishable differences exist in the frequencies beyond the fundamental frequency of the synchronisation signal; however, as the fundamental frequency dominates other frequency components, in terms of relative power, these differences are often obscured. Applying a bandpass filter to the reference signal and signal samples minimizes the influence of the fundamental frequency on the filter response by removing that portion of the signal altogether.

We chose to use a Fourier Transform based 'box filter', as the output produced by Butterworth and Chebyshev type filters, particularly those of a high order, proved highly irregular due to the high sampling rate used in acquiring the signals and comparatively narrow bandwidth of the filters. Nonetheless, it may be advantageous to pursue alternative bandpass filters in order to avoid the possible transient affects introduced by this method of filtering. To 'apply' the bandpass filter, the Fast Fourier Transform was applied to the input signal, $X(\omega) = \mathscr{F}\{x(t)\}$, to obtain amplitude and phase values for the frequency components. The filtering operation is defined as

$$X(\omega) = \begin{cases} X(\omega), & 2\pi f_h \leq X(\omega) \leq 2\pi f_l \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

where f_l and f_h give the lowpass and highpass frequency cutoffs for the bandpass filter, respectively. The inverse Fast Fourier Transform was then used to bring the signal back into the time domain.

Through experimentation, by use of several bandpass filters with increments of 1 MHz in

bandwidth, we have determined that, for some devices, the 13–16 MHz frequency range exhibits the greatest variation. We disregard frequencies higher than 20 MHz due to that the fact that the power levels of these components approach that of the noise floor in our testbed.

4.2.3.3 Normalisation

The Euclidean norm of the signal $x(t)$ is denoted by $\frac{x(t)}{\parallel x(t) \parallel}$, where

$$\parallel x(t) \parallel = \sqrt{\int_{-\infty}^{+\infty} x^2(t) \mathrm{d}t} \tag{4.6}$$

Normalising both the reference signal and signal being tested, according to the Euclidean norm, proves advantageous for discriminating between signals that are otherwise different but which, nonetheless, produce the same filter output due to the sum-of-products nature of the matched filter. If the filter output for two signals differs, however, this form of normalisation may decrease our ability to distinguish between the two, as the operation can be seen as desensitizing the matched filter to variations in amplitude.

4.2.3.4 Trimming

Time domain trimming was investigated in order to minimize the affect of the signal amplitude on filter response; by eliminating amplitude dominance, variations in the shape of the signal are made apparent. Analogous to the frequency domain trimming used in bandpass filtering, time domain trimming removes the portions of a signal that tend to overshadow all others by, for example, zeroing the signal amplitude for values greater than a predetermined lower bound. Furthermore, by adding an upper bound, and varying the height of each boundary accordingly, a window is created that allows for any portion of the signal to be scrutinised by its shape alone. For an arbitrary signal, $x(t)$, the trimming operation is defined as

$$x(t) = \begin{cases} 0, & l_{lower} \leq |x(t)| \leq l_{upper} \\ x(t), & \text{otherwise} \end{cases} \tag{4.7}$$

where l_{lower} and l_{upper} are the amplitude thresholds; the operation is symmetric about the t-axis.

For instance, to ensure that the width of a signal matches that of the filter, the zero-crossings—where the signal crosses the t-axis—could be examined by setting a lower bound. We have found that time-domain trimming is most effective when only the signal samples are trimmed, not the filter.

4.2.4 Using the signal profile

To determine whether or not a record originated from the device it claims to represent, the output of applying that device's filter to the record must fall within the current thresholds set for the device. More precisely, consider a record, r_{i^*}, that has been captured from a device asserting the identity of the i^{th} device. The record will be accepted as genuine so long as the following constraint is met

$$th^i_- \leq \max(\mathrm{f}_i(t) \star \mathrm{r}_{i^*}(t)) \leq th^i_+ \tag{4.8}$$

where $\mathrm{f}_i(t)$ is the i^{th} device's matched filter and $th^i_{+/-}$ the upper/lower limits, for the point in time at which the frame was captured, of its filter output. The reason for the use of the max function is given in Section 4.3.2. If the pre-processing techniques outlined above are used, the record r_{i^*} should be prepared appropriately and the corresponding filter used in (4.8).

4.3 Experimental approach

The equipment and methods used to acquire the Ethernet signals for analysis are given. Methods for calculating the false-accept rate (FAR) and false-reject rate (FRR) are discussed along with details about evaluating the efficacy of the matched filter approach. In order to determine the consistency, or lack thereof, of device behaviour over time, data was taken using the same cards at three different times, using two different experimental setups. Based upon these data, possible forensic application of PLIS are discussed.

4.3.1 Data collection

Two testbeds were used for the collection data. Each setup consisted of two PCs: one to act as the Test PC (TPC), which housed the Ethernet card to be fingerprinted, while the

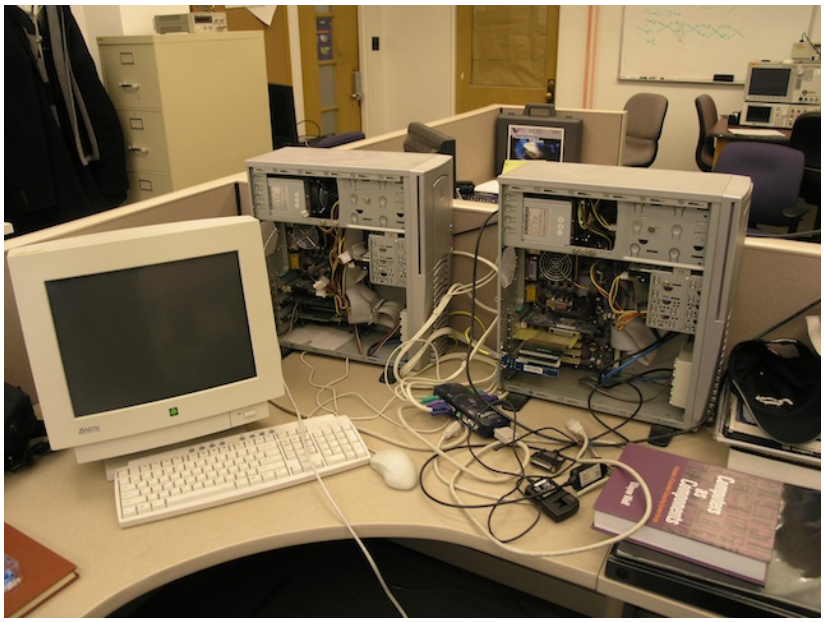

Figure 4.4 Experimental setup for dataset1: (left) DAQPC and (right) TPC; Tektronix 3052 DSO oscilloscope not shown.

other, the Data Acquisition PC (DAQPC), made use of a passively tapped internal Ethernet card to capture Ethernet frames sent to it over a crossover cable by the TPC. The earlier setup (circa 2003, Figure 4.4) ran GNU/Linux on both PCs and made use of a Tektronix 3052 digital sampling oscilloscope (DSO), interfaced via an IEEE 488 card and Labview-6, to acquire frames, while the later setup (circa 2010, Figure 4.5, 4.6) used a Tektronix 4032 digital phosphor oscilloscope (DPO), interfaced via USB and controlled by MATLAB (see Appendix A for the source of the acquisition routine). The first setup was used in the collection of the data comprising *dataset1* and the second setup was used for *dataset2* and *dataset3*.

In order to generate traffic for the DAQPC to capture, the TPC was instructed to ping the DAQPC. During a typical data acquisition period the TPC would ping the DAQPC 10,000 times over the course of approximately three hours. To ensure that only traffic from the TPC was captured and that the measurement equipment did not affect the load characteristics of the DAQPC, as seen by the TPC, only the receiving pins of the DAQPC's Ethernet card on the secondary side of the transformer were connected to the oscilloscope. In this way the DAQPC could respond to the TPC's pings and ensure that the data acquisition process didn't cause

Figure 4.5 Experimental setup for dataset2/3: (left) DAQPC and (right) TPC; oscilloscope below DAQPC (partial view).

packet loss or affect the transmitting circuitry of the TPC.

Upon detection of an Ethernet frame (a simple slope-based threshold was used) the oscilloscope began to sample the signal at a rate of 1 Gigasamples/s for dataset1 and 2.5 Gigasamples/s for dataset2/3. For the first dataset, the signal was sampled 10,000 times, for a total of 10 micro-seconds, while for the second and third datasets the signal was sampled 1,000,000 times, for a total of 400 micro-seconds. Both oscilloscopes had 8-bits of resolution; however, because of the DSO could store relatively fewer sample points than the DPO, dataset1 contained only the synchronisation signal and the beginning of the MAC destination address, while dataset2/3 retained the entire Ethernet frame. Practically, this meant that more of the Ethernet frame could be used in the construction of matched filters and more tests run on datasets two and three than on dataset one.

Finally, the data collected during sampling was sent to the DAQPC via an IEEE 448 or USB interface, where a custom Labview or MATLAB routine monitoring the interface accepted the data and stored the values in a vector called a record, which was subsequently written to disc. Each captured frame was stored in its own record; all of the records collected for a device

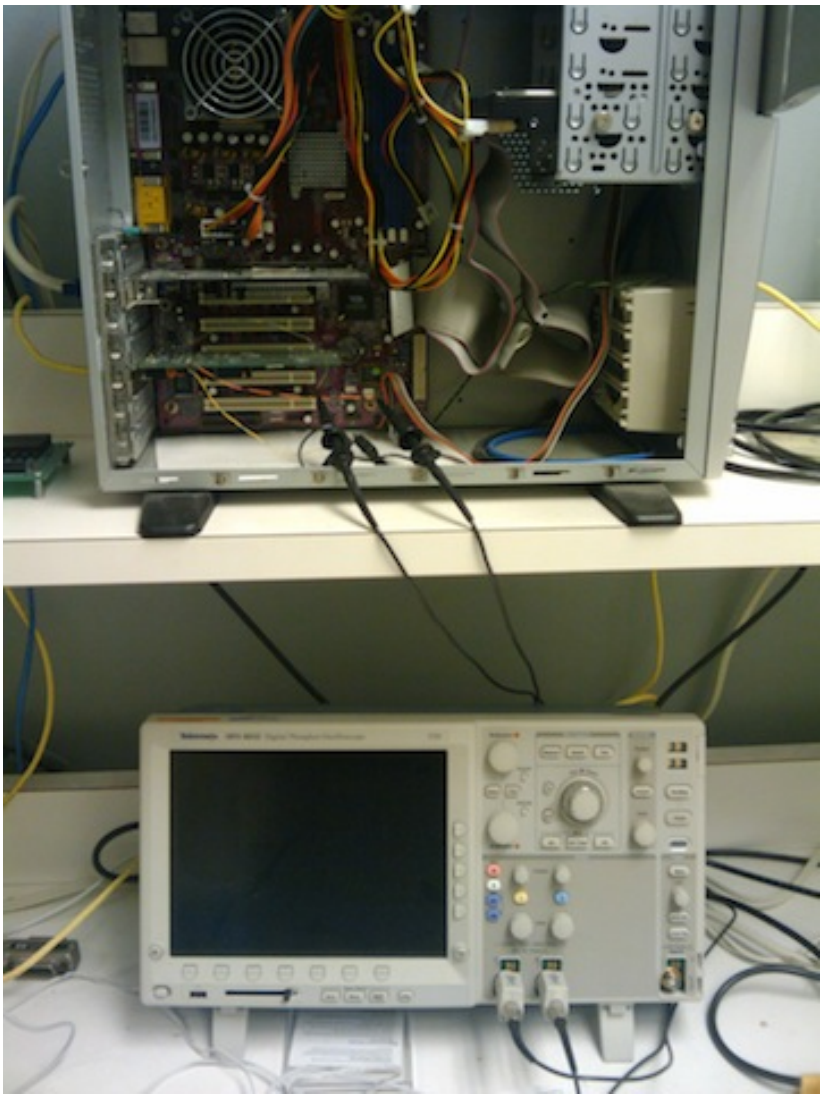

Figure 4.6 Oscilloscope connected to DAQPC for dataset2/3: (top) DAQPC and (bottom) Tektronix 4032 DPO oscilloscope. The oscilloscope is connected to the receive pins on the secondary side of the DAQPC's transformer; the ground clip of each probe is connected to the common ground of the transformer IC.

during a session are said to encompass its dataset.

The technical and procedural aspects of this process, for dataset1 (though the procedure is much the same, differing where noted above, for dataset2/3), are covered in greater detail by Jackson [20]. In what follows, unless explicitly stated, references to the data apply to all three datasets.

4.3.2 Filter application

Having acquired several thousand signal samples from each device over a number of hours, a filter was then created for each of the devices using the procedure outlined in Section 4.2.2.1. The reference signal for each device was selected from, and the operation appropriate to the variation performed upon, the first valid record of the device's dataset. In the case of the generic matched filter, for example, the reference signal had a period of roughly 6,000 ns and spanned roughly the length of the synchronisation signal. Following this, the reference signal was convolved with each record of its dataset, each of which may or may not have had the same variational operation performed upon it, using an FFT-based convolution algorithm. Convolving a reference signal with a record performs the matched filter operation for all possible time-shifts; consequently, an output is created that is equal in length to that of the length of the record. This operation was necessary, as opposed to the simpler one given by (4.4), because the t_0 of the reference signal may not in fact provide optimal alignment between the reference and test signals—and hence not produce the maximum filter output—due to the fact that an oscilloscope is not guaranteed to always trigger at the same point of the Ethernet frame. This might occur due to triggering error or, indeed, signal variation, and is unavoidable as the setup lacked an external trigger—i.e. the synchronised clock mentioned in Section 4.1—and instead relied on simple level-slope triggering. Thus, the filter output at the point of actual best alignment, $\gamma(t_a)$, was taken to correspond to the maximum of the convolution operation, which should have occurred at or near the original t_0[1]. Having determined the filter output for each record of its own dataset, the filter is then applied, again using convolution, to each record

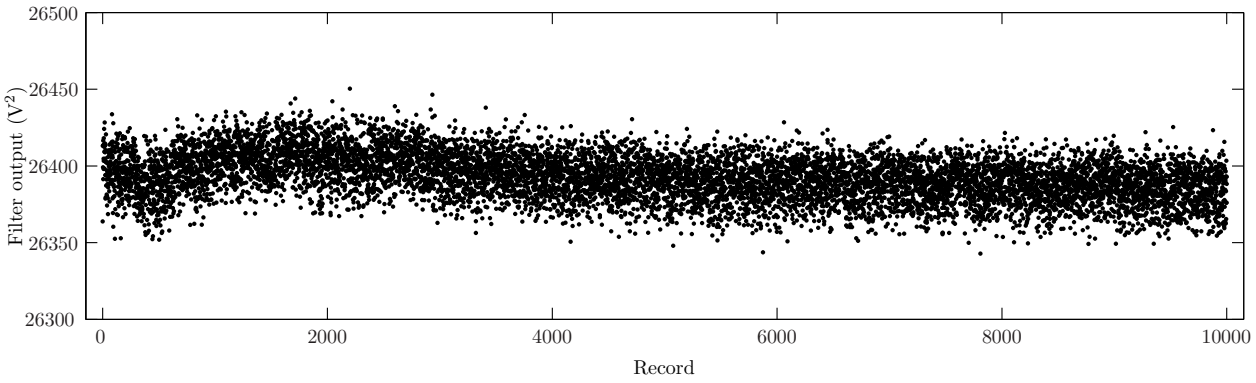

Figure 4.7 Control filter output, $c_i^j(t_a)$, for 10,000 records of an Ethernet device (*cve* 0.0011).

of the other device's datasets in order to determine the alikeness of their respective signals.

More precisely, allowing $f_i(t)$ to represent the reference signal for the i^{th} device and $r_i^j(t)$ the j^{th} record of its dataset, the filter output, $c_i^j(t_a)$, is

$$c_i^j(t_a) = \max(f_i(t) \star r_i^j(t)) \text{ for } j = 1 \cdots n \tag{4.9}$$

where n is the number of records in the device's dataset (Figure 4.7). This procedure was followed for each device ($i = 1 \cdots m$, where m is the number of devices) in order to obtain the filter response of each record in its dataset—the so-called *control* response.

The filter output or *subject* response, $s_{i,k}^j(t_a)$, of the k^{th} device using the i^{th} device's filter is

$$s_{i,k}^j(t_a) = \max(f_i(t) \star r_k^j(t)) \text{ for } j = 1 \cdots n \tag{4.10}$$

Equation 4.10 is used for devices $k = 1 \cdots m, k \neq i$ in order to find which devices could be differentiated from device i. For example, Figure 4.8 gives the filter outputs of two devices (*Device 1*, the control, and *Device 2*, the subject) using a filter derived form the first device's dataset. Following the explanation set forth in Section 4.2.2.2, as the filter outputs do not overlap, the matched filter PLIS is therefore able to discriminate between *Device 1* and *Device 2*.

[1] In point of fact, the triggering of the oscilloscope proved reliable enough—i.e. either signal did not vary significantly or trigger jitter was sufficiently low—for dataset2/3 that the area of the signal considered for taking the maximum between was reduced to ±50 samples around t_0 (on average the maximum filter output occurred within 1.563 sample points of t_0, in the case of dataset2). As such, Equations 4.9, 4.10, found below, should be reinterpreted accordingly.

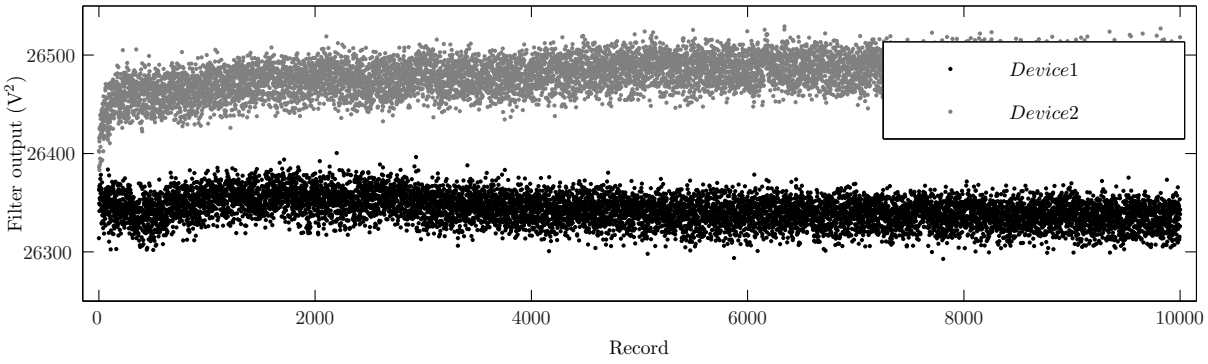

Figure 4.8 Filter output for 10,000 records of two different Ethernet devices using the same filter.

Comment on time complexity

Using an FFT-based convolution in applying the matched filter to an input signal requires two FFTs and one inverse-FFT, to wit

$$\gamma(t_a) = \max\left(\mathscr{F}^{-1}\left\{\mathscr{F}\{\alpha(t_0 - t)\} \cdot \mathscr{F}\{\beta(t)\}\right\}\right) \tag{4.11}$$

where $\alpha(t_0 - t)$ and $\beta(t)$ are defined as in Section 4.1. Each of these three transforms is of $O(n \lg n)$ complexity. It becomes readily apparent that using such an approach with the pre-processing techniques outlined in Section 4.2.3, which require multiple applications of the filter, will become computationally infeasible in anything close to real-time. It is possible, however, to reduce the matched filter operation to $O(n)$ complexity for subsequent applications of the filter after a single, initial use of the convolution approach.

Returning to the discussion of the definition of the matched filter in Section 4.1, it is seen that by (4.4) the matched filter operation is reduced to an inner-product of two signals, and thus linear complexity, if the signals are aligned. Letting $\Delta = t_a - t_0$ be the difference between the actual sample point of maximum alignment (determined through convolution) and the sample point (stipulated by the filter transfer function) that should produce maximum alignment. The filter output is then determined by

$$\gamma(t_a) = \int_{t_a - T}^{t_a} \alpha(\tau - \Delta)\beta(\tau)\mathrm{d}\tau \tag{4.12}$$

which is simply to say that the reference signal must be shifted forwards or backwards according to the difference between the expected alignment point, t_0, and the actual alignment point, t_a, before the inner-product is taken.

4.3.3 Threshold calculation

To summarise the discussion on the imposition of limits for, and tracking of, the filter output in Section 4.2.2.3: in order to account for changes in signal characteristics over time, the affects of which correspond to the variability in filter output seen in Figure 4.7, thresholds were introduced for the maximum amount of deviation in filter response acceptable before a signal is deemed too different from the original. Past behaviour must also be considered when setting these thresholds, and so it is required that the next m-frames resemble the previous n-frames, with an allowance for some variability.

If the distribution (along with its location and scale) of the filter output is known, it is trivial to calculate arbitrary confidence intervals with statistical rigour to bound future outputs. However, as all of these parameters must be estimated, *prediction intervals* must be used to build thresholds with any statistical basis [51]. By the use of a two-sided prediction interval, it is possible to state, with $100(1-\alpha)\%$ confidence, that all m future outputs will fall within the range of

$$th_{+/-}(\mathrm{c}^{j}\cdots\mathrm{c}^{j+m-1}) = \mu(\mathrm{c}^{j-n}\cdots\mathrm{c}^{j-1}) \pm r_{(1-\alpha;m,n)} \times \sigma(\mathrm{c}^{j-n}\cdots\mathrm{c}^{j-1}) \tag{4.13}$$

where c^{j} is the filter output for j^{th} record of the control device, and μ and σ are the mean and standard deviation, respectively, of that output. A 'conservative approximation' for r is given by

$$r_{(1-\alpha;m,n)} \approx \left(1+\frac{1}{n}\right)^{\frac{1}{2}} t_{(1-\alpha/(2m);n-1)} \tag{4.14}$$

where t is the *Student's t-distribution*. For the experiments performed the filter output for the first 25 frames of a device were used as training data to predict the output of the next 20 ($n = 25$, $m = 20$); specifying $\alpha = 0.05$ to achieve 95% certainty estimates yields $r = 3.397$. It was found that large, sporadic deviations do occur for all Ethernet devices, so a perfect acceptance rate cannot be obtained unless one is willing to allow a certain number of significant deviations

every m-frames or set r unreasonably high. As with any system with statistical variation, a balance must be found for α that results in acceptable false positives and false negatives.

While (4.13, 4.14) are meant for normal data, and, strictly speaking, by use of the *Lilliefors* test, it was observed that the filter output is not consistently normal (it exhibits a marked tendency towards the extreme value distribution), as a practical matter it is sufficiently close to a normal distribution as to obviate the need to apply a normal-transform (the Box-Cox transformation, e.g.) or make use of alternate prediction methods (such as ARMA modelling [54]).

4.3.4 Classifying filter output

The methodology used to evaluate the efficacy—i.e., the false positive and false negative counts—of the matched filter approach is set forth. For ease of interpretation, in what follows 'false reject' (a device not recognised as itself) and 'false accept' (a device misidentified as another) are occasionally made use of to mean 'false positive' and 'false negative', respectively.

4.3.4.1 Type I Errors

To determine the number of false positives for the i^{th} control device, the first 25 of its filter responses, $\mathrm{c}_i^{1\cdots 25}(t_a)$, were used in conjunction with the procedure set forth in the previous section to establish thresholds, which were expected to provide a false reject rate of 1% or less, for the next 20 outputs. If the filter response for one of the next 20 records lay outside of the bounds set by these thresholds, its corresponding record was marked as rejected and was not used in determining the thresholds for the next 20 outputs. It should be noted that the last five filter outputs used to calculate thresholds for the preceding 20 were used again for the next 20, as 25 outputs are needed for training while only 20 are accepted per iteration. This procedure was followed for the remainder of the filter responses in the device's dataset. The false positive count was then simply $FP = n_c - n - n_r$, where n_r is the total number of rejected records, n the number of records used for training, and n_c the total number of records for the device.

4.3.4.2 Type II Errors

Whereas it was possible to determine false rejects by sequentially applying (4.13) to each of the next 20 filter outputs, false accepts cannot be determined in such a sequential manner, as it cannot be known where to begin comparing the output of the i^{th} device's filter applied to the k^{th} device's dataset and the i^{th} device's own output in order to find their intersection. Simply comparing the distributions of the filter outputs for the two cases—by use of the overlapping coefficient, say—would also produce an inaccurate false negative count, as the filter output for each device was non-stationary and therefore not amenable to distribution fitting over the long term.

Therefore, to calculate the number of false accepts accurately, it was assumed that the filter response for each record of the k^{th} subject device's dataset using the i^{th} control device's filter, $\mathrm{s}_{i,k}^{1\cdots n_s}(t_a)$, where n_s is the number of records in the subject's dataset, was equally likely at all points in time. More familiarly, an attacker had the presumed ability to select an arbitrary record from the subject dataset and substitute it for a control record at will. The question, then, is how many records, on average, would an attacker be able to pass off as the control device during any 20 record period? To determine this, threshold values were calculated for $\mathrm{c}_i^{26\cdots 45}(t_a)$ (the first 25 records of $\mathrm{c}_i(t_a)$ as training data and consequently have no thresholds) to check how many filter outputs of $\mathrm{s}_{i,k}^{1\cdots n_s}(t_a)$ were erroneously accepted as the filter responses for records 26–45 of the control device. This procedure was followed for each subsequent 20 response segment of $\mathrm{c}_i^{46\cdots n_c}(t_a)$, leading to an average false negative count of

$$FN = \frac{m}{n_c} \sum_{l=1}^{n_c/m} n_a^l \tag{4.15}$$

where n_a^l is the number of records accepted per 20 output threshold period. Thus, if a preponderance of subject outputs were consistently misidentified as control outputs, the false negative count will be correspondingly high.

4.3.5 Combining tests

When coupled with the matched filter operation, each of the pre-processing techniques outlined in Section 4.2.3 produced an individual filter output, which could then be tracked

according to the threshold regime described above. By combining the results of the these filters, a multitude of what may be thought of as *tests*, which cover a broad feature set, could be used to establish the verisimilitude of the subject record. However, if multiple tests are to be utilised when deciding whether to accept or reject a record, a classifier combination technique must be used to reconcile divergent test results.

While many approaches to the problem of classifier combination are available [55], most were inappropriate for use in this context, as they required assumptions about the *a priori* probabilities of the inputs, which cannot be known or even estimated. Instead, the chosen approach for fixing upon a decision to accept or reject relied upon a bipartite approach: use statistical methods to set a lower bound on the number of tests a given record must pass—i.e., be inside the thresholds established for the individual filter output—before being accepted or, when the number of tests utilised was small, make use of a simple logical AND operation for their combination. While the former technique was not needed in the present case to produce good results, it is believed that the rationale for it, and the technique itself, might be of interest to other researchers working in this area.

The statistical bounds approach differs from the one set forth in Section 4.3.3 in that, whereas the distribution of the individual outputs is approximately normal, the distribution of the number of test failures per record could not be satisfactorily characterised using common distributions (due to the fact that the tests put forth above are self-evidently not independent, and would thus be expected to, and indeed did, exhibit complex, correlated relationships). As a consequence of this, one-sided distribution-free confidence bounds [51] are used. These bounds require greater amounts of training data to produce an acceptably high confidence bound and are not adaptive. Nevertheless, the technique could be used to calculate a conservative lower bound on the number of tests passed per record, with $100(1-\alpha)\%$ confidence, that at least $100p\%$ of future control records should exceed.

Allowing k to be a sorted list, from least to greatest, of the number of tests passed by the control records $\mathrm{c}^1 \cdots \mathrm{c}^n$. The lower bound for the number of tests each future record must pass

in order to be accepted is then $k(l)$, where 'l is chosen as the largest integer such that'

$$1 - \mathrm{B}(l-1; n, p) \geq 1 - \alpha \tag{4.16}$$

with B representing the Binomial distribution. It should be noted that while the number of records used as training data, n, should generally be as large as possible, a certain minimum does exist for which no value of l, for a given p and α, may be found to satisfy (4.16). This is actually slightly advantageous in that it obviates the need to make discretionary decisions about the size of the training data, as the minimum number of training records necessary to ensure that at least $100p\%$ of future records meet the lower bound for a certain confidence level can be exactly calculated by increasing n until a solution to (4.16) is found.

Having established the threshold for the minimum number of tests to be passed by the control device, type I errors and type II errors could then be determined by applying the methodology of Section 4.3.4 to each of the individual tests, with the added step of ensuring that the total number of tests passed per record exceeds said threshold. Which is to say, accept a record as originating from the control device so long as $n_p \geq k(l)$, where n_p is the number of filter outputs within the thresholds given by (4.13).

4.4 Analysis of results

The results of the matched filter methodology for signal profiling are presented for dataset1 (16 devices) and dataset2/3 (27 devices), with approximately $10,000$ records per dataset, consisting of a combination of three different models (Table 4.1). The naming convention $mXcY$ is used to denote card Y of model X. Testing parameters are discussed and metrics indicating the overall effectiveness of both the individual and combined approaches are given. Any mention of sample points used in a test are made with reference to Figure 4.1 for dataset1 and Figure 4.2 for dataset2/3.

4.4.1 Variety and scope of tests

The aim of this work was two fold: to establish the value of a generic matched filter approach to device identification and to measure the usefulness of the variations set forth in Section 4.2.3.

Table 4.1 Details of Ethernet cards used for experiments (dataset1: m4c1–3, m5c1–10, m6c1–3; dataset2/3: all).

Manufacturer/Model	Identifier	MAC Address	Serial	Chipset Markings	Dataset
D-Link/DFE-530TX+ (Rev. E1)	m4c1	00:40:05:34:a0:31	B229237077076	DL10038D, 33098Q1, 315F	1,2,3
	m4c2	00:40:05:36:01:15	B229237077139	DL10038D, 33246Q1, 316F	1,2,3
	m4c3	00:40:05:36:01:19	B229237077140	DL10038D, 33246Q1, 316F	1,2,3
	m4c4	00:40:05:35:75:40	B229237077075	DL10038D, 33098Q1, 315F	2,3
	m4c5	00:40:05:34:a0:30	B229237077074	DL10038D, 33098Q1, 315F	2,3
	m4c6	00:40:05:36:01:1a	B229237077133	DL10038D, 33246Q1, 316F	2,3
Genica/GN-788	m5c1	00:00:e8:12:65:36	DB0211105319	0206TABEDC2736.00	1,2,3
	m5c2	00:00:e8:12:17:db	DB0211105339	"	1,2,3
	m5c3	00:00:e8:12:2c:85	DB0211105358	"	1,2,3
	m5c4	00:00:e8:12:61:53	DB0211105396	"	1,2,3
	m5c5	00:00:e8:12:6d:77	DB0211105389	"	1,2,3
	m5c6	00:00:e8:12:61:47	DB0211105364	"	1,2,3
	m5c7	00:00:e8:12:65:2e	DB0211105349	"	1,2,3
	m5c8	00:00:e8:12:c4:a0	DB0211105317	"	1,2,3
	m5c9	00:00:e8:12:61:09	DB0211105326	"	1,2,3
	m5c10	00:00:e8:12:32:4a	DB0211105404	"	1,2,3
	m5c11	00:00:e8:12:65:3e	DB0211105394	"	2,3
Netronix/37NB-12290-311 (Rev. 1.1)	m6c1	00:08:54:0c:37:5f	122901133CF05938	VT6105, 0325cd, 23B4002200	1,2,3
	m6c2	00:08:54:0c:37:13	122901133CF05997	VT6105, 0325cd, 23B4001100	1,2,3
	m6c3	00:08:54:0c:37:4c	122901133CF05948	VT6105, 0326cd, 23B4401200	1,2,3
	m6c4	00:08:54:0c:37:42	122901133CD05949	VT6105, 0325cd, 23B4401100	2,3
	m6c5	00:08:54:0c:37:10	122901133CF06000	VT6105, 0326cd, 23B4401200	2,3
	m6c6	00:08:54:0c:37:55	122901133CF05939	VT6105, 0326cd, 23B4401200	2,3
	m6c7	00:08:54:0c:37:54	122901133CF05940	VT6105, 0325cd, 23B4002200	2,3
	m6c8	00:08:54:0c:37:0f	122901133CF05999	VT6105, 0326cd, 23B4401200	2,3
	m6c9	00:08:54:0c:4c:bf	122901133CF06650	VT6105, 0325cd, 23B4001100	2,3
	m6c10	00:08:54:0c:37:4d	122901133CF05947	VT6105, 0325cd, 23B4002200	2,3

To this end, a total of 526 tests were devised, the precise nature of which will be described shortly, and then applied to the datasets. A complete analysis of the available data required that each test be carried out on every one of the $\sim 10,000$ records in each device's datasets and that each device be tested against every other device using itself as a reference. Unless specified otherwise, each test used the same portion of the synchronisation signal (475–6650 ns for dataset1 and 1310–7315 ns for dataset2/3) as a reference signal before the requisite processing occurred.

4.4.1.1 Bandpass filtering

A total of 210 filters were used to exhaustively test the bandwidth of 0–20 MHz at 1 MHz increments (Table 4.2). As the power of the frequency components of the synchronisation signal beyond 20 MHz were slight (due to the presence of a 17 ± 1 MHz low-pass filter on the DAQPC's Ethernet card), it was decided that extending the bandwidth beyond this range would contribute little as these small contributions to the filter output would be necessarily overshadowed by the inner product operation. Both the reference signal and test signal were filtered.

Table 4.2 Bandwidths of filters used in BPF pre-processing.

High-pass freq. (MHz)	Low-pass freq. (MHz)				
19	20				
18	19	20			
⋮	⋮	⋮	⋱		
1	2	3	⋯	20	
0	1	2	3	⋯	20

4.4.1.2 An ensemble of filters

It was hypothesised that when a signal undergoes significant change so too does the circuitry used to produce the signal; this in turn may produce unique transients. As such, additional filters were used to examine areas of abrupt change in the synchronisation signal (all times approximate): the transient and synchronisation signal (300–6650 ns for dataset1 and 1200–7315 ns for dataset2/3), the transient only (300–475 ns for dataset1 and 1200–1310 ns for dataset2/3), the transition from the synchronisation signal to destination MAC address (6650–6830 ns for dataset1 and 7315–7500 ns for dataset2/3), and the transient with the synchronisation signal and transition to the MAC address (300–6830 ns for dataset1 and 1200–7500 ns for dataset2/3). For dataset2/3 the synchronisation signal with the transition to the MAC address was also tested (1310–7500 ns). Because the entire frame was captured for dataset2/3, it was possible to extend this analysis to several other areas of the signal: the entirety of the destination MAC address and part of the source address (7500–21515 ns), the transition to the MAC addresses with the destination and source addresses (7315–21515 ns), the synchronisation signal and the transition to the MAC addresses with the destination and source addresses (1310–21515 ns), and the entirety of the data from the transient to the end of the portion of the source address (1200–21515 ns).

4.4.1.3 Normalisation

The Euclidean norm was applied to the aligned test and reference signals in each test. As such, there were only 263 unique tests; the remainder were simply the output of the original test divided by the product of the norm of the reference and aligned test signals.

4.4.1.4 Trimming

Amplitude trimming, using increments of 0.25 volts from zero to three volts, was used in two ways. *Lower*-trimming set the data values below a give level to zero, while *upper*-trimming set the values above said level to zero. More precisely, for an input signal, $x(t)$, and $l = 0.25, 0.50, \ldots, 3.0$ volts, lower-trimming was defined as

$$x(t) = \begin{cases} 0, & |x(t)| \leq l \\ x(t), & \text{otherwise} \end{cases} \tag{4.17}$$

while for upper-trimming

$$x(t) = \begin{cases} 0, & |x(t)| \geq l \\ x(t), & \text{otherwise} \end{cases} \tag{4.18}$$

These trimming procedures were applied to the test signal individually in one instance and both the reference and test signals in another.

4.4.2 Results

Several metrics common to machine learning and data mining [56], as well as confusion matrices, are used to present the results of the matched filter approach to device differentiation. Both forms of presentation are necessary, as metrics provide only an overall picture of the performance for a single device, while confusion matrices show the degree (the amount of overlap between two devices) and direction (whether one device was being confused with another and vice versa) of specific instances of misclassification.

Before proceeding with an explanation of the metrics used and the format of the confusion matrices, it should be noted that, in the context of this work, a true positive (TP) is understood to be a record rightly rejected as not having originated from the control device, while a false positive (FP) is a control record wrongly rejected as not having originated from the control. A true negative (TN) is then a rightly accepted record that originated from the control, while a false negative (FN) is a wrongly accepted record that did not originate from the control. Additionally, the false negatives and true positives for a given control were calculated with respect to all of the subject devices; i.e. the individual FN and TP counts of the subject

devices against a particular control were summed to determine the overall false-negative and true-positive counts for each control.

The *accuracy* ($\mathrm{A} = (TP+TN)/(TP+TN+FP+FN)$) *precision* ($\mathrm{P} = TP/(TP+FP)$), *recall* ($\mathrm{R} = TP/(TP+FN)$), and *specificity* ($\mathrm{S} = TN/(TN+FP)$) metrics are used to give a reasonable account—at least when the metrics are interpreted collectively—of the overall effectiveness of the classification system. Note: specificity is equivalent to the true-negative rate of the confusion matrix.

Confusion matrices, on the other hand, are employed to provide a finer view of how well the methodology performed in distinguishing control devices from individual subject devices by juxtaposing the true-negative rates for each of the controls against the false-negative rates of the subjects. In the matrices, control devices occupy the rows and subject devices the columns so that true-negative rates appear along the diagonal with false-negative rates in off-diagonal elements. Each row thus shows the percentage of the time the method was able to correctly identify the control and differentiate it from the subjects. A perfect classifier would produce a matrix where diagonal elements are one and off-diagonal ones are zero.

4.4.2.1 Generic matched filter

The confusion matrix and APRS values for the generic matched filter are presented in Tables 4.3–4.5 and 4.6–4.8, respectively. Thresholds based on (4.13) produce overly wide intervals for the filter output, and as such were established for at most a 5% FPR ($\alpha = 0.05$ results in $r = 3.397$; see Equation 4.14) to achieve the desired FPR of less than 1%. As can be seen from Tables 4.3–4.5, the TNR was sufficiently high (greater than 99%) while, at least for different model cards, the FNR was nearly 0% for dataset1 and 0% for dataset2/3, though some cards of the same model were difficult to differentiate. We attribute 0% inter-model FNR for dataset2/3 to the fact the time at which the oscilloscope triggered on the signal was used to constrain the search for the maximum filter output (see the footnote in Section 4.3.2) for these datasets.

While the method failed to discriminate between devices of the same model in all cases (as evidenced by the individual recall rates; especially for the m4 cards), the overall accuracy ($\sim$90% for dataset1 and $\sim$94%/$\sim$93% for dataset2/3) was satisfactory considering that all of

the cards used in the analysis were purchased in bulk, at the same time, with the majority most likely originating from the same manufacturing lot (the serial numbers and chipset markings given in Table 4.1 would seem to support this).

The choice to use such a sample was a conscious one, in that we were attempting to examine a worst-case scenario. Even so, we would like to note that the lack of overlap between different model cards results in a matrix that is relatively sparse. Combined with the fact that a very large number of records were used in the analysis, almost perfect APRS values would be produced if inter-model results were used; however, these would also be perfectly irrelevant in our scenario. Accordingly, we have calculated APRS values for the intra-model case only. The results and their method of calculation, when taken together, imply that in a diverse environment—i.e., one in which a great number of differing model devices are present—the generic matched filter would perform well at differentiating devices from one another.

It may not even be strictly necessary for a network to be composed of heterogeneous devices: so long as the cards are sufficiently different from each other for the approach to work, the device population could consist of same-model cards. We had hoped that distances between MAC addresses and/or serial numbers would correlate negatively to the FNR, in which case we might then be able to establish a minimum distance between these identifiers to guarantee distinguishability. While this is possibly the case for the m4 cards (the sample size is too small to draw definite conclusions), it does not hold for either of the m5 or m6 cards, as can be seen by comparing the FNRs of these cards with the information found in Table 4.1. Perhaps cards not examined in this work do, in fact, correlate in such a way. In any case, without assistance from manufacturers, we must attempt to infer differences in manufacture time (and implicitly behaviour) from the cards themselves. Also, we have, admittedly, not carried out a proper sample survey, so these results should be regarded as preliminary at best. Note, however, that an effective survey would likely require an auxiliary variable that is correlated with the FNR to decrease the variance of the population parameter [57].

Changes to the data collection regime for dataset2/3 improved the performance of the generic matched filter. Comparing Tables 4.7, 4.8 to 4.6 shows that increasing the rate at which the synchronisation signal is sampled to 2.5 Gigasamples/s reduces the collisions of the

m4 cards (1–3) and m6 cards (1–3) used in dataset1 to zero, though other collisions do arise for the remaining devices in these model classes in dataset2/3.

[RESULTS FOLLOW]

Table 4.3 Confusion matrix for the generic matched filter (dataset1)

	Subject															
	m4			m5										m6		
Control	c1	c2	c3	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c1	c2	c3
m4c1	0.999	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c2	0	0.999	0.891	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c3	0	0.955	0.998	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c1	0	0	0	0.999	0.991	0.001	0	0	0.008	0.001	0.102	0	0	0	0	0
m5c2	0	0	0	0.951	0.999	0	0	0	0.012	0	0.136	0	0	0	0	0
m5c3	0	0	0	0.001	0	0.999	0	0	0	0.993	0	0	0	0	0	0
m5c4	0	0	0	0	0	0	0.998	0	0	0	0	0.001	0.001	0	0	0
m5c5	0	0	0	0	0	0	0	0.999	0	0	0	0	0	0	0	0
m5c6	0	0	0	0.007	0.015	0	0	0	0.998	0	0.906	0	0.713	0	0	0
m5c7	0	0	0	0.001	0	0.986	0	0	0	0.998	0	0	0	0	0	0
m5c8	0	0	0	0.073	0.167	0	0	0	0.953	0	0.999	0	0.289	0	0	0
m5c9	0	0	0	0	0	0	0	0	0	0	0	0.998	0	0	0	0
m5c10	0	0	0	0	0	0	0.002	0	0.547	0	0.258	0	0.998	0	0	0
m6c1	0	0	0	0	0	0	0	0	0	0	0	0	0	0.997	0.333	0
m6c2	0	0	0	0	0	0.001	0	0	0	0	0	0	0	0.306	0.997	0.069
m6c3	0	0	0	0	0	0.321	0	0	0	0.500	0	0	0	0	0.057	0.997

Table 4.4 Confusion matrix for the generic matched filter (dataset2)

	Subject																									
	m4						m5											m6									
Control	c1	c2	c3	c4	c5	c6	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
m4c1	0.999	0	0.009	0	0	0.491	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c2	0	0.999	0	0.765	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c3	0.001	0	0.996	0	0	0.155	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c4	0	0.739	0	0.999	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c5	0	0	0	0	0.999	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c6	1.000	0.017	0.999	0	0	1.000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c1	0	0	0	0	0	0	0.996	0.923	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c2	0	0	0	0	0	0	0.993	0.998	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c3	0	0	0	0	0	0	0	0	0.999	0	0	0	0	0.001	0	0	0.332	0	0	0	0	0	0	0	0	0	0
m5c4	0	0	0	0	0	0	0	0	0	0.996	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c5	0	0	0	0	0	0	0	0	0	0	0.999	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c6	0	0	0	0	0	0	0	0	0	0	0	0.996	0.913	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c7	0	0	0	0	0	0	0	0	0	0	0	0.966	0.998	0	0	0.004	0	0	0	0	0	0	0	0	0	0	0
m5c8	0	0	0	0	0	0	0	0	0.069	0	0	0	0	0.998	0	0	0.885	0	0	0	0	0	0	0	0	0	0
m5c9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.997	0	0	0	0	0	0	0	0	0	0	0	0
m5c10	0	0	0	0	0	0	0	0	0	0	0	0	0.002	0	0	0.997	0	0	0	0	0	0	0	0	0	0	0
m5c11	0	0	0	0	0	0	0	0	0.368	0	0	0	0	0.641	0	0	0.999	0	0	0	0	0	0	0	0	0	0
m6c1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.997	0	0	0	0	0	0	0.155	0	0.860
m6c2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.998	0	0	0	0	0	0	0	0
m6c3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.996	0	0	0	0	0	0	0
m6c4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.996	0	0	0	0	0	0
m6c5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.997	0	0	0	0	0
m6c6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.999	0.851	0	0	0
m6c7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.337	0.996	0	0	0
m6c8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.538	0	0	0	0	0	0	0.999	0	0.643
m6c9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.999	0
m6c10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.000	0	0	0	0	0	0	0.633	0	0.999

Table 4.5 Confusion matrix for the generic matched filter (dataset3)

	Subject																									
	m4						m5											m6									
Control	c1	c2	c3	c4	c5	c6	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
m4c1	0.999	0	0.008	0	0	0.496	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c2	0	0.999	0	0.721	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c3	0.234	0.906	0.992	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c4	0	0.747	0	1.000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c5	0	0	0	0	0.999	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c6	1.000	0.014	1.000	0	0	1.000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c1	0	0	0	0	0	0	0.997	0.941	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c2	0	0	0	0	0	0	0.977	0.998	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c3	0	0	0	0	0	0	0	0	0.999	0	0	0	0	0.012	0	0	0.438	0	0	0	0	0	0	0	0	0	0
m5c4	0	0	0	0	0	0	0	0	0	0.997	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c5	0	0	0	0	0	0	0	0	0	0	0.999	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c6	0	0	0	0	0	0	0	0	0	0	0	0.996	0.921	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c7	0	0	0	0	0	0	0	0	0	0	0	0.953	0.998	0	0	0.004	0	0	0	0	0	0	0	0	0	0	0
m5c8	0	0	0	0	0	0	0	0	0.051	0	0	0	0	0.998	0	0	0.864	0	0	0	0	0	0	0	0	0	0
m5c9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.000	0	0	0	0	0	0	0	0	0	0	0	0
m5c10	0	0	0	0	0	0	0	0	0	0	0	0	0.002	0	0	0.997	0	0	0	0	0	0	0	0	0	0	0
m5c11	0	0	0	0	0	0	0	0	0.551	0	0	0	0	0.912	0	0	0.999	0	0	0	0	0	0	0	0	0	0
m6c1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.997	0	0	0	0	0	0	0.148	0	0.858
m6c2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.998	0	0	0	0	0	0	0	0
m6c3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.996	0	0	0	0	0	0	0
m6c4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.996	0	0	0	0	0	0
m6c5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.997	0	0	0	0	0
m6c6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.999	0.852	0	0	0
m6c7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.344	0.996	0	0	0
m6c8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.522	0	0	0	0	0	0	0.999	0	0.640
m6c9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.999	0
m6c10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.000	0	0	0	0	0	0	0.628	0	0.999

Table 4.6 Intra-model APRS values for the generic matched filter (dataset1)

Tested Card	A	P	R	S
m4c1	1.000	0.999	1.000	0.999
m4c2	0.702	0.999	0.554	0.999
m4c3	0.681	0.998	0.522	0.998
m5c1	0.890	1.000	0.877	0.999
m5c2	0.890	1.000	0.878	0.999
m5c3	0.900	1.000	0.890	0.999
m5c4	1.000	1.000	1.000	0.998
m5c5	1.000	1.000	1.000	0.999
m5c6	0.836	1.000	0.818	0.998
m5c7	0.901	1.000	0.890	0.998
m5c8	0.852	1.000	0.835	0.999
m5c9	1.000	1.000	1.000	0.998
m5c10	0.919	1.000	0.910	0.998
m6c1	0.888	0.998	0.834	0.997
m6c2	0.874	0.998	0.813	0.997
m6c3	0.980	0.998	0.971	0.997
mean	**0.895**	**0.999**	**0.862**	**0.998**

Table 4.7 Intra-model APRS values for the generic matched filter (dataset2)

Tested Card	A	P	R	S
m4c1	0.918	1.000	0.902	0.999
m4c2	0.872	1.000	0.846	0.999
m4c3	0.974	0.999	0.969	0.996
m4c4	0.876	1.000	0.852	0.999
m4c5	1.000	1.000	1.000	0.999
m4c6	0.663	1.000	0.597	1.000
m5c1	0.916	1.000	0.908	0.996
m5c2	0.910	1.000	0.901	0.998
m5c3	0.970	1.000	0.967	0.999
m5c4	1.000	1.000	1.000	0.996
m5c5	1.000	1.000	1.000	0.999
m5c6	0.917	1.000	0.909	0.996
m5c7	0.912	1.000	0.903	0.998
m5c8	0.913	1.000	0.905	0.998
m5c9	1.000	1.000	1.000	0.997
m5c10	1.000	1.000	1.000	0.997
m5c11	0.908	1.000	0.899	0.999
m6c1	0.898	1.000	0.887	0.997
m6c2	1.000	1.000	1.000	0.998
m6c3	1.000	1.000	1.000	0.996
m6c4	1.000	1.000	1.000	0.996
m6c5	1.000	1.000	1.000	0.997
m6c6	0.915	1.000	0.905	0.999
m6c7	0.966	1.000	0.963	0.996
m6c8	0.882	1.000	0.869	0.999
m6c9	1.000	1.000	1.000	0.999
m6c10	0.837	1.000	0.819	0.999
mean	**0.935**	**1.000**	**0.926**	**0.998**

Table 4.8 Intra-model APRS values for the generic matched filter (dataset3)

Tested Card	**A**	**P**	**R**	**S**
m4c1	0.917	1.000	0.901	0.999
m4c2	0.879	1.000	0.855	0.999
m4c3	0.808	0.998	0.771	0.992
m4c4	0.875	1.000	0.850	1.000
m4c5	1.000	1.000	1.000	0.999
m4c6	0.663	1.000	0.597	1.000
m5c1	0.914	1.000	0.906	0.997
m5c2	0.911	1.000	0.902	0.998
m5c3	0.959	1.000	0.955	0.999
m5c4	1.000	1.000	1.000	0.997
m5c5	1.000	1.000	1.000	0.999
m5c6	0.916	1.000	0.908	0.996
m5c7	0.913	1.000	0.904	0.998
m5c8	0.917	1.000	0.908	0.998
m5c9	1.000	1.000	1.000	1.000
m5c10	1.000	1.000	1.000	0.997
m5c11	0.867	1.000	0.854	0.999
m6c1	0.899	1.000	0.888	0.997
m6c2	1.000	1.000	1.000	0.998
m6c3	1.000	1.000	1.000	0.996
m6c4	1.000	1.000	1.000	0.996
m6c5	1.000	1.000	1.000	0.997
m6c6	0.915	1.000	0.905	0.999
m6c7	0.965	1.000	0.962	0.996
m6c8	0.884	1.000	0.871	0.999
m6c9	1.000	1.000	1.000	0.999
m6c10	0.837	1.000	0.819	0.999
mean	**0.927**	**1.000**	**0.917**	**0.998**

4.4.2.2 Combined approach

Though initially devised to provide increased efficacy over the generic matched filter, most of the tests proposed in Section 4.2.3 were unable to better our ability to distinguish between devices. Even so, the actual work of carrying out an analysis of all the tests was useful in that it identified the areas of the signal common to all devices—with respect to the matched filter—and made apparent those techniques and domains which should be explored further.

Exempting those tests that were effective for a single pair of cards, only the bandpass filters (usually using non-normalised records) provided some advantage over the generic matched filter. Interestingly, the bandwidth of the these filters differed by model; i.e., while a particular bandpass filter may have been effective for, say, the m5 cards it could not be depended upon to give good results for the other models. This would seem to imply, in the best case, that different model cards exhibit model-specific deviations, with tendencies towards variation being roughly constant across the model. Of course, this hypothesis should be tested by carrying out a proper survey to ensure that the tests deemed effective are not merely artefacts of *ex post facto* selection.

While the number of effective tests proved few, their use in combination—i.e., each record was required to pass all of the tests to be accepted (a.k.a unanimous voting [55])—showed a marked improvement over the results obtained in Section 4.4.2.1, with average accuracies of ∼95% for dataset1, and ∼100% for dataset2/3 (Tables 4.9–4.11 and 4.12–4.14); specificity declined slightly but still averaged over 98% for all datasets. It should be noted that the increased accuracy for dataset1 is largely due to the m5 cards, while for dataset2/3 cards of all models showed improvement.

dataset1

With the exception of the m6 cards, multiple bandpass filters were found to be at least partially effective for each model—though the bandwidths of the filters overlapped in each case. We utilised non-normalised and normalised filters (in the 6–16 MHz and 10–16 MHz ranges, respectively) for the m5 cards but made use of only one normalised filter for the m4 cards (0–12

MHz). It is significant that each of the m5 filters—and indeed all of the effective bandpass filters for this model—excluded the fundamental frequency of the 5 MHz synchronisation signal, where the majority of the power lies, but contained the frequencies near to, and including, 15 MHz, which is where the first harmonic of a 5 MHz square wave resides. This suggests that the greatest signal deviations are to be found at the higher harmonics; however, as noted earlier, due to the presence of a low-pass filter on the DAQPC, it was not possible for us to examine these harmonics. The Ethernet card in the DAQPC is a 10Mb one, so it is possible that 100Mb cards do not use filters with such a low cut-off frequencies, even in 10Mb mode. If this is the case, we may be able to improve our results by examining the harmonics, individually or in combination, past 15 MHz.

dataset2/3

We hypothesised that, considering that the synchronisation signal is short in duration, relative to the rest of the frame, and as the performance of the filter generally increases as the signal it operates on is lengthened, the portion of the frame containing the source and destination MAC addresses could be used, along with the synchronisation signal, to eliminate overlap between devices. To this end the source address was set (spoofed) to be the same for each device in dataset2/3. Unfortunately, neither of the MAC addresses when used singly, in combination, or with the synchronisation signal—normalised or not—were able to differentiate more devices than the generic matched filter. While it is possible that a completely passive implementation of the matched filter PLIS could also utilise the source IP, these experiments with the MAC addresses indicate that individual device behaviour is expressed primarily in the synchronisation signal. It is of course possible that tests other than an ensemble of filters (e.g. bandpass filtering) would reveal differences in other parts of the frame.

The effective tests for dataset2/3, much like dataset1, proved to be bandpass filters. For models m4 and m6 normalised and non-normalised filters with bandwidths of 0–10MHz proved efficacious, while for m5 a 15–16MHz non-normalised filter worked best—these results conform broadly to those for dataset1, at least with respect to the bandwidths of the filters if not their normalisation or non-normalisation. For the 0–10MHz filters, the only component visibly

different over the bandwidth was the DC component. We take this to mean that the amplitudes of the differential signal differ by a consistent amount for each sample point; i.e. it is almost as though one channel or the other were an exact copy of the other but attenuated/amplified.

Finally, we speculate that the reason the 6–16 MHz filter was ineffective for dataset2/3 (the effective filters of dataset2/3 otherwise overlap to some extent with those of dataset1) is due to a difference between the data analysis routines used on dataset1 and dataset2/3. For dataset1 the filter analysis was carried out using the entire record, which meant that a portion of the 10 MHz MAC address portion of the Ethernet frame was tested, while for dataset2/3 only that portion of the synchronisation signal, which contains very little power near 10 MHz, aligned to the purported device's matched filter was used.

[RESULTS FOLLOW]

Table 4.9 Confusion matrix for combined matched filters (dataset1)

	Subject															
	m4			m5										m6		
Control	c1	c2	c3	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c1	c2	c3
m4c1	0.987	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c2	0	0.994	0.874	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c3	0	0.687	0.990	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c1	0	0	0	0.993	0.052	0	0	0	0.001	0.001	0	0	0	0	0	0
m5c2	0	0	0	0.003	0.993	0	0	0	0.011	0	0	0	0	0	0	0
m5c3	0	0	0	0	0	0.991	0	0	0	0.008	0	0	0	0	0	0
m5c4	0	0	0	0	0	0	0.993	0	0	0	0	0	0	0	0	0
m5c5	0	0	0	0	0	0	0	0.990	0	0	0	0	0	0	0	0
m5c6	0	0	0	0.004	0.015	0	0	0	0.989	0	0	0	0.023	0	0	0
m5c7	0	0	0	0	0	0.013	0	0	0	0.991	0	0	0	0	0	0
m5c8	0	0	0	0	0	0	0	0	0	0	0.991	0	0	0	0	0
m5c9	0	0	0	0	0	0	0	0	0	0	0	0.990	0	0	0	0
m5c10	0	0	0	0	0	0	0	0	0.094	0	0	0	0.992	0	0	0
m6c1	0	0	0	0	0	0	0	0	0	0	0	0	0	0.995	0.313	0
m6c2	0	0	0	0	0	0	0	0	0	0	0	0	0	0.290	0.996	0.064
m6c3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.052	0.996

Table 4.10 Confusion matrix for combined matched filters (dataset2)

	Subject																									
	m4						m5											m6									
Control	c1	c2	c3	c4	c5	c6	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
m4c1	0.991	0	0.004	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c2	0	0.991	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c3	0.001	0	0.987	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c4	0	0	0	0.983	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c5	0	0	0	0	0.984	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c6	0	0	0	0	0	0.992	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c1	0	0	0	0	0	0	0.989	0.005	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c2	0	0	0	0	0	0	0	0.988	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c3	0	0	0	0	0	0	0	0	0.993	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c4	0	0	0	0	0	0	0	0	0	0.989	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c5	0	0	0	0	0	0	0	0	0	0	0.989	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c6	0	0	0	0	0	0	0	0	0	0	0	0.985	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c7	0	0	0	0	0	0	0	0	0	0	0	0	0.990	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c8	0	0	0	0	0	0	0	0	0	0	0	0	0	0.991	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.983	0	0	0	0	0	0	0	0	0	0	0	0
m5c10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.988	0	0	0	0	0	0	0	0	0	0	0
m5c11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.991	0	0	0	0	0	0	0	0	0	0
m6c1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.976	0	0	0	0	0	0	0	0	0
m6c2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.982	0	0	0	0	0	0	0	0
m6c3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.975	0	0	0	0	0	0	0
m6c4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.974	0	0	0	0	0	0
m6c5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.976	0	0	0	0	0
m6c6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.978	0.029	0	0	0
m6c7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.018	0.974	0	0	0
m6c8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.977	0	0.002
m6c9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.976	0
m6c10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.004	0	0.980

Table 4.11 Confusion matrix for combined matched filters (dataset3)

	Subject																									
	m4						m5											m6									
Control	c1	c2	c3	c4	c5	c6	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
m4c1	0.990	0	0.003	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c2	0	0.991	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c3	0.229	0	0.988	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c4	0	0	0	0.984	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c5	0	0	0	0	0.984	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c6	0	0	0	0	0	0.992	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c1	0	0	0	0	0	0	0.992	0.003	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c2	0	0	0	0	0	0	0.004	0.989	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c3	0	0	0	0	0	0	0	0	0.991	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c4	0	0	0	0	0	0	0	0	0	0.990	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c5	0	0	0	0	0	0	0	0	0	0	0.990	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c6	0	0	0	0	0	0	0	0	0	0	0	0.985	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c7	0	0	0	0	0	0	0	0	0	0	0	0	0.991	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c8	0	0	0	0	0	0	0	0	0	0	0	0	0	0.991	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.983	0	0	0	0	0	0	0	0	0	0	0	0
m5c10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.988	0	0	0	0	0	0	0	0	0	0	0
m5c11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.989	0	0	0	0	0	0	0	0	0	0
m6c1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.978	0	0	0	0	0	0	0	0	0
m6c2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.981	0	0	0	0	0	0	0	0
m6c3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.975	0	0	0	0	0	0	0
m6c4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.974	0	0	0	0	0	0
m6c5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.976	0	0	0	0	0
m6c6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.978	0.028	0	0	0
m6c7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.017	0.976	0	0	0
m6c8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.977	0	0.002
m6c9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.978	0
m6c10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.003	0	0.980

Table 4.12 Intra-model APRS values for combined matched filters (dataset1)

Tested Card	A	P	R	S
m4c1	0.996	0.993	1.000	0.987
m4c2	0.706	0.994	0.562	0.994
m4c3	0.767	0.992	0.655	0.990
m5c1	0.994	0.999	0.994	0.993
m5c2	0.998	0.999	0.999	0.993
m5c3	0.998	0.999	0.999	0.991
m5c4	0.999	0.999	1.000	0.993
m5c5	0.999	0.999	1.000	0.990
m5c6	0.995	0.999	0.995	0.989
m5c7	0.998	0.999	0.999	0.991
m5c8	0.999	0.999	1.000	0.991
m5c9	0.999	0.999	1.000	0.990
m5c10	0.990	0.999	0.990	0.992
m6c1	0.894	0.997	0.843	0.995
m6c2	0.880	0.998	0.822	0.996
m6c3	0.981	0.998	0.974	0.996
mean	**0.950**	**0.998**	**0.927**	**0.992**

Table 4.13 Intra-model APRS values for combined matched filters (dataset2)

Tested Card	A	P	R	S
m4c1	0.998	0.998	0.999	0.991
m4c2	0.999	0.998	1.000	0.991
m4c3	0.998	0.997	1.000	0.987
m4c4	0.997	0.997	1.000	0.983
m4c5	0.997	0.997	1.000	0.984
m4c6	0.999	0.998	1.000	0.992
m5c1	0.999	0.999	1.000	0.989
m5c2	0.999	0.999	1.000	0.988
m5c3	0.999	0.999	1.000	0.993
m5c4	0.999	0.999	1.000	0.989
m5c5	0.999	0.999	1.000	0.989
m5c6	0.999	0.999	1.000	0.985
m5c7	0.999	0.999	1.000	0.990
m5c8	0.999	0.999	1.000	0.991
m5c9	0.998	0.998	1.000	0.983
m5c10	0.999	0.999	1.000	0.988
m5c11	0.999	0.999	1.000	0.991
m6c1	0.998	0.997	1.000	0.976
m6c2	0.998	0.998	1.000	0.982
m6c3	0.998	0.997	1.000	0.975
m6c4	0.997	0.997	1.000	0.974
m6c5	0.998	0.997	1.000	0.976
m6c6	0.995	0.998	0.997	0.978
m6c7	0.996	0.997	0.998	0.974
m6c8	0.998	0.997	1.000	0.977
m6c9	0.998	0.997	1.000	0.976
m6c10	0.998	0.998	1.000	0.980
mean	**0.998**	**0.998**	**1.000**	**0.984**

Table 4.14 Intra-model APRS values for combined matched filters (dataset3)

Tested Card	**A**	**P**	**R**	**S**
m4c1	0.998	0.998	0.999	0.990
m4c2	0.999	0.998	1.000	0.991
m4c3	0.960	0.998	0.954	0.988
m4c4	0.997	0.997	1.000	0.984
m4c5	0.997	0.997	1.000	0.984
m4c6	0.999	0.998	1.000	0.992
m5c1	0.999	0.999	1.000	0.992
m5c2	0.999	0.999	1.000	0.989
m5c3	0.999	0.999	1.000	0.991
m5c4	0.999	0.999	1.000	0.990
m5c5	0.999	0.999	1.000	0.990
m5c6	0.999	0.999	1.000	0.985
m5c7	0.999	0.999	1.000	0.991
m5c8	0.999	0.999	1.000	0.991
m5c9	0.998	0.998	1.000	0.983
m5c10	0.999	0.999	1.000	0.988
m5c11	0.999	0.999	1.000	0.989
m6c1	0.998	0.998	1.000	0.978
m6c2	0.998	0.998	1.000	0.981
m6c3	0.997	0.997	1.000	0.975
m6c4	0.997	0.997	1.000	0.974
m6c5	0.998	0.997	1.000	0.976
m6c6	0.995	0.998	0.997	0.978
m6c7	0.996	0.997	0.998	0.976
m6c8	0.997	0.997	1.000	0.977
m6c9	0.998	0.998	1.000	0.978
m6c10	0.998	0.998	1.000	0.980
mean	**0.997**	**0.998**	**0.998**	**0.984**

4.4.2.3 Stability and forensics

In comparing dataset2 and dataset3 (Tables 4.4, 4.5 and 4.10, 4.5) we see that devices that were difficult to differentiate in one dataset showed similar overlap in the other. As dataset2/3 were taken one month apart, this suggests that, at least over the short term, device behaviour is stable. With regards to differences in collisions between dataset1 and dataset2/3: it may be that the device behaviour changed enough in the years between when the data were acquired to shift the incidences of collision (e.g., from m5c6 vs. m5c8 in dataset1 to m5c6 vs. m5c7 in dataset2/3); however, sufficient documentation about dataset1 does not exist to conclusively state that the card labels of dataset1 actually refer to the cards with the same labels in dataset2/3 (that is, card m5c8 in dataset1 might not be the same as card m5c8 in dataset2/3, for example).

The preceding should not be taken to imply that device identity can be established in the present using past behaviour (i.e. thresholds for a device's filter output the next day or week cannot be calculated using present outputs), but merely that devices that were distinguishable in the recent past will continue to be so in the near term.

Determining whether or not device identity is consistent in the long term or, for example, whether a device can be reauthenticated after having lost and reestablished its connection to the network, is not only essential if PLI is to be utilised for network authentication and/or intrusion detection, but is of particular importance for forensic applications as well. For instance, if the Ethernet frames related to an attack on a network are captured, and a suspected device later obtained, the frames could be compared to test frames generated by the device using the PLIS to determine whether or not it was likely that the device perpetrated the attack (of course we would have to know something about the uniqueness of device behaviour for the population of such devices, with respect to the PLI being used in the comparison, so that some qualifier, like there is a 1 in 1000 chance that another device shares the same fingerprint, could be given).

With this application in mind, we used the generic filters created for each device of dataset2 against all of the devices in dataset3. Tables 4.15 and 4.16 present the results of this analysis. The 'control' response given along the diagonal was calculated in the same way as the subject

response (see Section 4.3.4.2). With reference to the APRS metrics, we are first concerned with specificity: for if a device can't be identified as itself nothing can actually be known about how well a PLI does in differentiating other devices from it. Specificity varied by model, and even greatly within a model class. The m4 cards proved more difficult to re-identify than cards from the other models, m5 cards were re-identified over 94% of the time, on average, while the m6 cards showed mixed results with five devices over 87% but three devices less than 50%. Because of the low TNR rates for m4 devices, the average accuracy was only ∼90%. Taken together, this implies that, at least for certain models (e.g., the m5 model, which had an average specificity and accuracy of over 94%), forensic analysis could be effected legitimately.

Forensics and long-term tracking of devices are an open problem in PLI. For the matched filter two possible approaches would be: increased retention of past filter outputs and sub-profiles based upon different operating environments/conditions. In the first case, a device's signal profile could be endowed with a type of memory so that filter outputs could be put into a historical context; i.e. given what we have seen recently from the device—taking into account the last time we heard from the device—does the current filter output fit the profile? For the second, profiles could be built for how a device behaves at different times of of day or under varying CPU loads, etc. Filter outputs for newly arrived frames could be compared against these different profiles to see if a match is found and if it's a likely match, given the device's reported load and the time of day, etc.

[RESULTS FOLLOW]

Table 4.15 Confusion matrix for dataset2 generic matched filters used on dataset3

	Subject																										
	m4						m5											m6									
Control	c1	c2	c3	c4	c5	c6	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
m4c1	0	0.595	0	0	0	0.170	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c2	0	0	0	0.899	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c3	0.052	0.242	0.003	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c4	0	0	0	0.971	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c5	0	0	0	0	0.259	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m4c6	0.378	1.000	0.604	0	0	1.000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c1	0	0	0	0	0	0	0.768	0.885	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c2	0	0	0	0	0	0	0.838	0.991	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c3	0	0	0	0	0	0	0	0	0.968	0	0	0	0	0.032	0	0	0.321	0	0	0	0	0	0	0	0	0	0
m5c4	0	0	0	0	0	0	0	0	0	0.979	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c5	0	0	0	0	0	0	0	0	0	0	1.000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c6	0	0	0	0	0	0	0	0	0	0	0	0.975	0.655	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c7	0	0	0	0	0	0	0	0	0	0	0	0.977	0.939	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m5c8	0	0	0	0	0	0	0	0	0.026	0	0	0	0	0.986	0	0	0.886	0	0	0	0	0	0	0	0	0	0
m5c9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.980	0	0	0	0	0	0	0	0	0	0	0	0
m5c10	0	0	0	0	0	0	0	0	0	0	0	0.001	0.008	0	0	0.824	0	0	0	0	0	0	0	0	0	0	0
m5c11	0	0	0	0	0	0	0	0	0.133	0	0	0	0	0.941	0	0	0.991	0	0	0	0	0	0	0	0	0	0
m6c1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.431	0	0	0	0	0	0	0	0	0.130
m6c2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.934	0.019	0	0.002	0	0	0	0	0
m6c3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.001	0.628	0	0	0	0	0	0	0
m6c4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.727	0	0	0	0	0	0
m6c5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.197	0	0	0	0	0
m6c6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.999	0.615	0	0	0
m6c7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.441	0.874	0	0	0
m6c8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.975	0	0	0	0	0	0	0.062	0	0.994
m6c9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.001	0	0	0	0	0.976	0
m6c10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.985	0	0	0	0	0	0	0	0	0.877

Table 4.16 Intra-model APRS values for for dataset2 generic matched filters used on dataset3

Tested Card	**A**	**P**	**R**	**S**
m4c1	0.706	0.809	0.847	0.000
m4c2	0.684	0.804	0.820	0.000
m4c3	0.785	0.825	0.941	0.003
m4c4	0.995	0.994	1.000	0.971
m4c5	0.877	0.871	1.000	0.259
m4c6	0.670	1.000	0.604	1.000
m5c1	0.898	0.975	0.911	0.768
m5c2	0.923	0.999	0.916	0.991
m5c3	0.965	0.997	0.965	0.968
m5c4	0.998	0.998	1.000	0.979
m5c5	1.000	1.000	1.000	1.000
m5c6	0.938	0.997	0.934	0.975
m5c7	0.906	0.993	0.902	0.939
m5c8	0.916	0.999	0.909	0.986
m5c9	0.998	0.998	1.000	0.980
m5c10	0.983	0.983	0.999	0.824
m5c11	0.902	0.999	0.893	0.991
m6c1	0.930	0.940	0.986	0.431
m6c2	0.991	0.993	0.998	0.934
m6c3	0.963	0.960	1.000	0.628
m6c4	0.973	0.971	1.000	0.727
m6c5	0.920	0.918	1.000	0.197
m6c6	0.938	1.000	0.932	0.999
m6c7	0.943	0.985	0.951	0.874
m6c8	0.709	0.882	0.781	0.062
m6c9	0.998	0.997	1.000	0.976
m6c10	0.889	0.985	0.891	0.877
mean	**0.904**	**0.958**	**0.933**	**0.716**

4.5 Conclusion

We presented a methodology capable of identifying Ethernet cards based upon minute variations in their network signalling resulting from hardware and manufacturing inconsistencies, using an optimal detector, the matched filter. Several non-traditional applications of the filter were presented in order to improve its ability to discriminate between signals from seemingly identical devices of the same manufacturing lot. The experimental results of applying these filters to three different models of Ethernet cards, totalling 27 devices, and over three different datasets, were presented and discussed. Our results indicate that a matched filter can easily discriminate between Ethernet cards of different models and, with sufficient pre-processing of data, cards of the same model to an acceptable degree of accuracy, and that device behaviour is stable enough for some models of devices to justify further research into forensic applications for PLI.

CHAPTER 5. DIFFERENCE SENSITIVITY

The entire basis of PLI rests upon the assertion that slight variations [of devices] are difficult, if not impossible, to control and duplicate (Section 1.2. In light of recent work [47, 48], which shows that wireless signals can be successfully forged, it is no longer sufficient to merely assert the inherent security of PLIS. A brief overview of these works is instructive as it provides information on not only how PLI can be attacked but also the hardware necessary to do so—both of these pieces of information are necessary when considering ways to determine the security of a PLIS.

Both [47, 48] consider two types of attacks against the PLIS proposed in [35], which utilised the demodulation characteristics of 802.11b signals; in addition, a transient-based approach for sensor nodes is examined in [47]. [35] was compromised in both works by creating signals with the features of known devices and through replay of observed frames. For the former attack, false-accept rates (FAR) of 98% and 75% were reported for [47, 48], respectively; in the latter attack, the FAR for [48] was 55% while the replay attack met with similar success as the generation attack for [47]. The difference in attack success rates can probably be attributed to not only the threat models but the vastly different hardware used to model the PLI system and carry out the attacks.

In [47] universal software radio peripherals (USRP) operating at 128 Megasamples/s and controlled with the GNU Radio library were used for both the genuine and attacker devices, with the attacker device being programmed to produce the features of the genuine devices as measured by, and at, the PLI system (which consisted of an Agilent Digital Signal Analyzer operating at 40Gigasamples/s with 8000MHz of bandwidth). The replay attack was carried out using a Tektronix AWG 7000 (20 Gigasamples/s); the frames used for the replay were captured at the attacker's location using the PLI system. In [48] both the PLI system and

the attacking device were built using the same USRP (14-bit analogue-to-digital converter operating at 100 Megasamples/s and dual 16-bit digital-to-analogue converter operating at 400MHz). The attacker sought to reproduce or generate signals, which it captured, from one of three laptops used to represent legitimate users.

In their analysis of [32], Danev *et al.* were able to successfully replay frames captured by the PLI system over a wired channel; however, when a wireless channel was used the system could only be defeated if the attacker assumed the genuine device's physical location.

Based upon the character of these attacks, we proposed a general framework for determining the theoretical limits of a given classification technique to distinguish between signals from different devices. Having found the maximum and minimum waveform voltage values necessary to distinguish two signals, it was shown that it is possible to define the characteristics of an arbitrary waveform generator (AWG) necessary to produce a forged signal that is acceptable to the PLIS. The work focused on studying the limits of the matched filter, though the framework is general enough to apply to other classification techniques (although its application may be more complicated).

5.1 Limitations of analysis

In what follows, all analysis and calculations are carried out with respect to a differential, or reconstructed, 10Mb Ethernet waveform (depicted in Figure 4.1), which is found by taking the difference of the signals captured at the receive pins on secondary side of the DAQPC's transformer. While this is unimportant in determining how different the signals from two devices must be before they are distinguishable, it does result in a loosening of the constraints placed on an attacker. In the former case this simplification is unimportant because the matched filter methodology assumes the reconstructed signal (we needn't be concerned with the constituent signals as only their difference is considered). However, for the latter case, the analysis ignores the fact that in actuality an attacker would be required to forge two signals. We have made this assumption not only because it simplifies the analysis by not requiring that we consider two signals and also their difference, but also because the only data available to us is the differential waveform and that acquiring the signals the attacker need forge would require that we take

measurements on the primary side of the DAQPC transformer.

In addition, we also ignore problems of alignment and the affects of the channel, from the perspective of an attacker. In the former case, this means that we take for granted that the attacker is able to create a signal that results in the correct alignment; while the latter implies that a measurement made at one point along the wire, or even a tap connected to the wire, would yield the same results as a measurement made at any other point at the same time. Again, this downplays the complexity of carrying out an attack but it also provides a worst case analysis of the PLIS. We have also only performed a first-order analysis; i.e., we do not comment on the interdependence of the AWG characteristics given in section 5.3.3. Finally, a note about notation: a record is taken to mean a sampled waveform from an Ethernet frame. When we speak of forging records, we mean that the waveform represented by the record would be forged.

5.2 Constraint on signal differences

As laid out in section 4.3, for device k to be accepted as device i the maximum of the convolution between the reference signal for the i^{th} device, $f_i(t)$, and the records of the k^{th} device, $r_k(t)$, must fall between the thresholds th_+ and th_-. To simplify the analysis, we will make use of discrete notation and view the matched filter in terms of correlation; i.e. we will dispense with defining the filtering operation in terms of convolution and instead use the dot product. As such, it is no longer necessary for the reference signal to be time-reversed.

In the most general case then—one that assumes no prior restrictions about signal alignment—a record from device k must satisfy the following constraint to be accepted by the matched filter PLIS as originating from device i

$$th_- \leq \max\left(\sum_{j=1}^{n} f_i[j] \times r_k[j+\Delta]\right) \leq th_+ \tag{5.1}$$

where n is the length of the reference signal for device i, m is the length of the record from device k, and Δ may vary from $0 \cdots m-n$.

Equation 5.1 tells us how different two signals must be until they are distinguishable. While this may be seen as merely another way of formulating the matched filter PLIS, it is done to

make explicit the relationship of the levels of the original signals to the thresholds used to decide whether two signals originated from the same device (this is necessary for the analysis that follows). In the case of other PLIS, similar work would need to be carried out, with the difference that signal levels would first need to be connected to feature sets and then those features to thresholds (in the matched filter PLIS the signal levels serve as the feature set).

From the perspective of an attacker, (5.1) can be satisfied in one of two ways: by attempting to create a high fidelity copy of the waveform from device i used to create $th_{+/-}$, which would guarantee that the attacker's frames fall with the thresholds (*Type I* attack), or by manipulating an existing signal so that its filter output, using f_i, falls with $th_{+/-}$ (*Type II* attack). Having formulated the signal-level constraint under which two devices may be considered identical, we can ask what sort of signal generation or manipulation capabilities, and information about the PLIS, an attacker would need to produce signals for each attack.

5.3 Type I attack

Allowing r_a to represent the attacker's records and r_g the records of the device the attacker is targeting (the genuine device), both sampled at the PLI system, in a type one attack an attempt is made to generate forged frames based upon genuine frames. Because the thresholds for the genuine device allow for the variation of the genuine device's filter output an attacker needn't produce perfect copies of the genuine device's waveform. Our task then is to determine how different the signal levels of r_a can be from r_g and still have the filter output of r_a fall within the thresholds established for r_g.

5.3.1 Threat model

For the type one attack forged frames could be passed off in one of two ways: having observed a frame the attacker could attempt to replay the synchronisation portion of the original waveform but with a different payload, or they could construct a single frame based upon the average of multiple observed waveforms and transmit it with a custom payload. If the attacker wants to maximise the amount of allowable error between the forged signals and the authentic signals they will choose the latter case. The proof follows.

Following the procedure set out in Section 4.3.3, the thresholds for the next m records are determined by taking the mean of filter outputs for the previous n records and adding, for the upper threshold, or subtracting, for the lower threshold, the standard deviation of those same outputs times some constant (see Equation 4.13). As the filter is the sum of products, forging a signal that produces the mean filter output allows for the maximum, equal amount of deviation for each sample point in either direction. The average of the signals used to calculate the thresholds is just such a signal.

Allowing s_g to represent those portions of $r_g^{i-n,\cdots,i-1}$ aligned to the reference signal f_g and l to be the length of f_g, the filter output for the j^{th} record ($j = i-n, \cdots, i-1$) is

$$c_g^j = \sum_{k=1}^{l} f_g[k] \times s_g^j[k] \tag{5.2a}$$

$$= f_g \cdot s_g^j \tag{5.2b}$$

The mean of the filter output for the n training records

$$\mu(c_g) = \frac{c_g^{i-n} + c_g^{i-n-1} +, \ldots, + c_g^{i-1}}{n} \tag{5.3a}$$

$$= \frac{f_g \cdot s_g^{i-n} + f_g \cdot s_g^{i-n-1} +, \ldots, + f_g \cdot s_g^{i-1}}{n} \tag{5.3b}$$

$$= \frac{f_g \cdot \left(s_g^{i-n} + s_g^{i-n-1} +, \ldots, + s_g^{i-1}\right)}{n} \tag{5.3c}$$

$$= f_g \cdot \mu\left(s_g\right) \tag{5.3d}$$

It is worth noting that although an infinite number of arbitrary signals (though not an infinite number of signals falling within the guidelines set by the 802.3 standard [58]) could be generated to produce a filter output equal to the mean of the previous n records, finding the average signal only requires that an attacker observe n waveforms, align, and then average them. Of course an attacker could not know the which frames would exactly constitute the n training records; nonetheless a sort of moving average using more than n waveforms could be employed to approximate the true mean as the filter output does not change quickly over time. (The exact limits of such an approximation could be at least partially estimated by calculating the rate of change of the width of the thresholds determined during the data analysis covered in Chapter 4.) Finally, while the attacker can align and average observed waveforms, there is

no guarantee that the resulting signal, even if reproduced perfectly, would be aligned to the genuine device's reference in such a way as to produce a filter output of $(th_+ - th_-)/2$.

Having determined the optimum signal an attacker would attempt to forge, we now turn to the question of much the attacker could deviate from the signal and still make it acceptable to the matched filter PLIS.

5.3.2 Signal deviation

In order to simplify the analysis, let us ignore for the moment the problem of alignment; i.e. following the triggering of the sampler used in the system, the next l samples are automatically used in the matched filter operation instead of trying to find that those sample points that result in maximum alignment. All of the above concerning the optimum signal to forge still holds as the average of these signals would still yield the average filter output. This simplification is really not so exceptional as it may seem because, as mentioned in Section 4.3.2, trigger jitter and signal variation proved so small in dataset2/3 that the maximum filter output was only sought within 50 sample points of the expected maximum point of alignment. (In point of fact, average deviation between the expected alignment, t_0, and the actual alignment, t_a, for a device tested against itself for dataset2 was only 1.563 sample points.) Allowing t_{trg} to denote the sample point in the at which the oscilloscope triggered, (5.1) may be rewritten as

$$th_- \leq \sum_{j=1}^{n} f_i[j] \times r_k[j + t_{trg}] \leq th_+ \tag{5.4}$$

The practical consequence of this, from the attacker's standpoint, is that the discrepancy between the attacker's average signal and the genuine average signal is greatly reduced, if not eliminated, as finding the average signal no longer requires pre-alignment to some reference signal but is instead carried out merely by an averaging of the records (this of course presumes comparable sampling equipment for the attacker and the identification system). In what follows we assume that the average signal computed (but not reproduced) by the attacker would be equal to that seen at the PLIS.

Allowing $s_{avg} = \mu(s_g)$, where s_g are the next l sample points following the trigger point of

r_g, the constraint equation the attacker needs to satisfy is then reduced to

$$th_- \leq \sum_{k=1}^{l} f_g[k] \times (s_{avg}[k] + A[k]) \leq th_+ \tag{5.5}$$

where $A[k]$ is the per sample point deviation of the attacker's *reproduced* averaged signal, s_a, from the genuine averaged signal (i.e. $A = s_a - s_{avg}$).

The maximum and minimum allowable values for $A[k]$ are found by increasing or decreasing the signal level for each sample point until the dot product of the modified signal and the reference signal equal the thresholds. Using an additive model for changes in signal levels leads to the following system of equations

$$th_{+/-} = \sum_{k=1}^{l} \begin{cases} f_g[k] \times \left(s_{avg}[k] + A_{+/-}\right) & \text{for} \quad s_{avg}[k] \geq 0 \\ f_g[k] \times \left(s_{avg}[k] - A_{+/-}\right) & \text{for} \quad s_{avg}[k] < 0 \end{cases} \tag{5.6}$$

where $A_+ \geq 0$ and $A_- \leq 0$. An additive model was chosen because it allows for an equal amount deviation for each sample point, no matter the signal level.

Another possible model is a multiplicative one in which more deviation is allowed for higher signal levels than lower ones

$$th_- \leq \sum_{k=1}^{l} f_g[k] \times \left(s_{avg}[k] \times A_{+/-}\right) \leq th_+ \tag{5.7}$$

where $0 \leq A_- \leq A_+$. This latter model would be more appropriate if an attacker's equipment had greater accuracy in producing smaller signal levels than higher ones. We assume that the attacker's capabilities are agnostic to signal level and therefore use the former model. This is in keeping with our conservative approach regarding the capabilities of an attacker's hardware (i.e. we want to provide an upper bound on AWG performance) and also accords with the characterisation of waveform generating equipment by manufacturers as their metrics are given so as to provide, if not uniform, then at least worst case indications of performance.

The solutions to (5.6) are given by

$$A_{+/-} = \frac{th_{+/-} - f_g \cdot s_{avg}}{\sum_{k=1}^{l} |f_g[k]|} \tag{5.8}$$

Thus, for the attacker so long as $A_- \leq A[k] \leq A_+$ the forged signal is guaranteed to be accepted as the genuine device by the matched filter PLIS. Because the thresholds $th_{+/-}$ are symmetric about the mean, $|A_+| = |A_-|$.

Having determined not only the signal an attacker should attempt to produce to have a forged record accepted, but also the how much the forged signal can vary from the ideal and still be accepted by the matched filter PLIS, we must ask what kind of AWG would be able to produce the ideal signal with an acceptable degree of accuracy (i.e. find the minimum performance required of an arbitrary waveform generator to produce the target signal within the boundaries established by (5.8).

5.3.3 Arbitrary waveform generator characterisation

An arbitrary waveform generator creates an analogue version of a digitized waveform. The three core components of an AWG are the waveform source memory, digital-to-analogue converter (DAC), and low-pass filter; optional components include scaling circuits, DC offset circuits, and differential outputs [59]. An analogue signal is created by feeding the binary values of the digitized waveform (known as codes) to the DAC, where a stepped, or discrete, output is generated; the stepped output is smoothed by the low-pass filter.

5.3.3.1 Parameters of interest

Because of the central role of the DAC in recreating the digital signal, we will concentrate our performance analysis exclusively on it and assume the other components of the AWG to be ideal. In any case, the parameters related to the DAC we will be discussing are always given with respect to the output of the AWG, so we are merely overestimating the minimum performance of the AWG. Additionally, for simplicity sake, we have not included the error bands associated with the waveform generator's output; an exact and full analysis of AWG performance would require that these bands be incorporated.

According to [60], the most important specifications used to evaluate the dynamic performance of a DAC are settling time, glitch impulse area, distortion, spurious free dynamic range (SFDR), and signal-to-noise ratio (SNR). Not all of these parameters are relevant or their impact calculable in the present case; in particular, glitch area will be subsumed under our discussion of settling time. In addition to these parameters, we will also discuss the impact of the resolution of the DAC. Finally, the algorithms described below are documented in Appendix

B.

Resolution

DAC resolution refers to the number of discrete outputs a converter is capable of generating [61]. A DAC with a resolution of n-bits is able to produce 2^n outputs within the range V_+ and V_- volts. Assuming linearly-spaced outputs, the voltage difference between two neighbouring outputs, which we shall refer to as the increment voltage, is $V_{FS}/(2^n - 1)$, where $V_{FS} = V_+ - V_-$ (the full-scale voltage). Analogue-to-digital converters used in sampling devices such as oscilloscopes also have a resolution, but in this case resolution refers to the number of signals levels they are capable of measuring.

Because s_{avg} is the average of n discrete signals, it may not be an exact multiple of the increment voltage of either the attacker's AWG or the PLIS system's sampling device. In actuality then the averaged signal an attacker would attempt to forge, s_{act}, would be made up of the closest multiples of the PLIS's increment voltage to the ideal s_{avg}. Allowing $m[i]$ to represent the multiple of the PLIS sampler's increment voltage leading to the smallest difference for the i^{th} point of s_{avg}, that is

$$\arg\min_{m} \left(\left| s_{avg}[i] - m\frac{V_{FS}}{2^n - 1} \right| \right) \text{ for } i = 1, \ldots, l \tag{5.9}$$

where V_{FS} and n are the full-scale voltage and resolution, respectively, of the sampler, then

$$s_{act} = m[i] \times \frac{V_{FS}}{(2^n - 1)} \tag{5.10}$$

The average signal measured at the PLI system would not necessarily give rise to a filter output equal to the mean of the previous n filter outputs, though it would approximate it.

An attacker need not actually require an AWG with a resolution equal to that of the PLI system's sampler to carry out a type one attack, as the attacker needn't actually generate s_{act} but merely a signal that when sampled satisfies Equation 5.6 (where s_{act} is substituted for s_{avg}). Allowing s_- to be the minimum signal and s_+ the maximum signal, calculated according to (5.8), that produces a filter output equal to th_- and th_+, respectively, the resolution, n_a, and full scale voltage, V^a_{FS}, of the attacker's must find some m between $0, \ldots, 2^{n_a} - 1$ for each

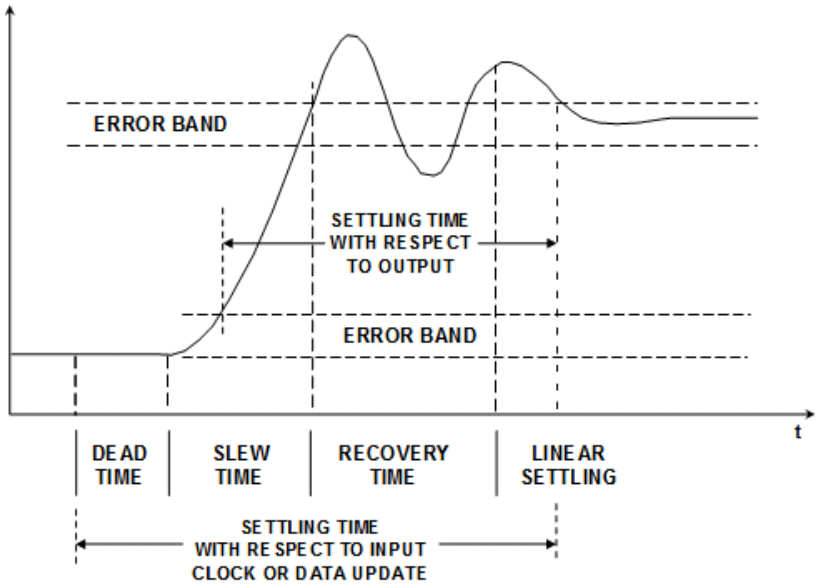

Figure 5.1 The components of settling time (Source: [62]).

$i = 1, \ldots, l$ such that

$$s_{-}[i] \leq m \times \frac{V_{FS}^{a}}{2^{n_a} - 1} \leq s_{+}[i] \tag{5.11}$$

In the analysis that follows, we assume that $s_{avg} = s_{act}$.

Setting time and glitch area

The settling time, τ_s, of a DAC refers to the time it takes the DAC to switch its output from one code and reach the steady-state value, within the error bands, of the new code [62]. As can be seen from Figure 5.1, the settling time can be further divided into: dead time (time it takes for the DAC to register the code change and begin changing output); slew time (time it takes for the DAC to first cross the error band of the new output); recovery time (time in which the DAC is acting in a non-linear manner—characterised as under- and over-shooting—that may cause the output to fall outside the error bands for the new output); and linear settling time (the time in which the DAC output approaches the error bands of the output in a linear fashion). Glitch area refers to the time integral of the signal falling outside the error bands during the recovery time (commonly defined for the worst-case code change).

Assuming that an attacker is attempting to reproduce s_{avg}, it may seem at first glance

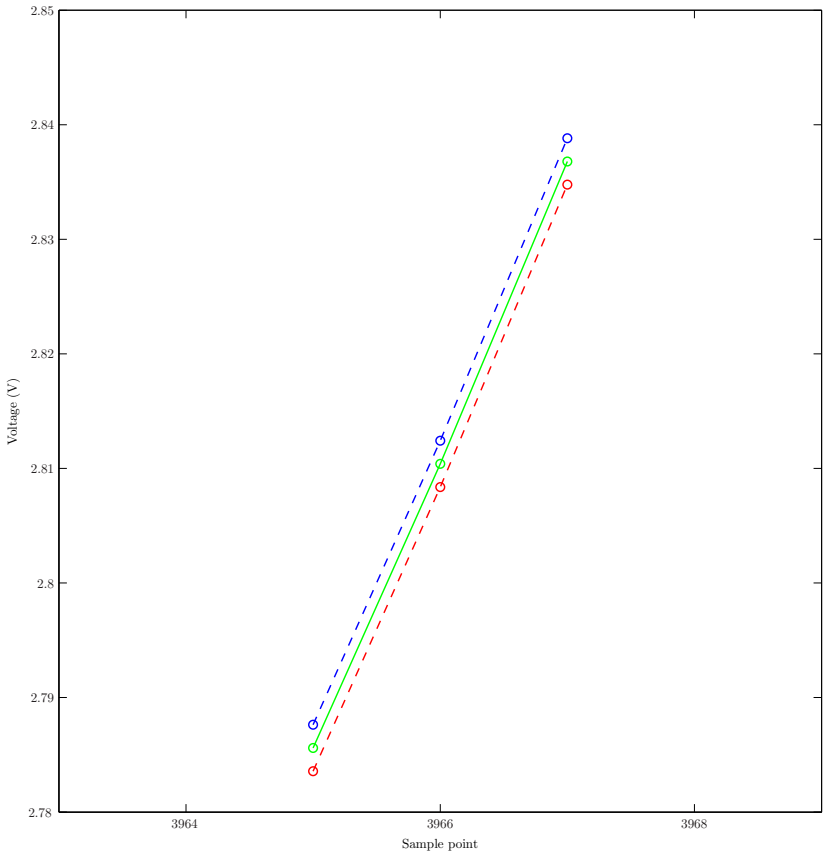

Figure 5.2 Settling time: optimal signal to use for attack, s_{avg} (green) with maximum, s_+, and minimum values, s_-, of attack signal (red). In moving from $s_{avg}[3396]$ to $s_{avg}[3397]$ an attacker only need reach $s_-[3397]$ by the next sampling period.

that an attacker's AWG must be able to change from to $s_{avg}[i]$ to $s_{avg}[i+1]$ before the PLIS samples the $i+1$ point; i.e. $\tau_s \leq 1/f_s$, where f_s is the sampling frequency of the PLIS oscilloscope. However, according to (5.6) an attacker need only reach an output that is within $s_{avg}[i+1] \pm A_{+/-}$ to ensure acceptance of the forged frame (Figure 5.2).

Specifically, allowing s_- to be defined as above, an attacker need only make sure that at each sampling point (i.e. every $1/f_s$ seconds) the output of the AWG be more than $s_-[j]/s_{avg}[j]$ of $s_{avg}[j]$ for the signal to be accepted as genuine (Figure 5.3).

In order to simplify the calculation that gives the settling time which satisfies this condition, we have linearised the slew and recovery times (see Figure 5.4). While this simplification provides a very conservative estimate of the minimum settling time, again in keeping with our general approach, it also obviates the need to consider the affects of glitch area in our analysis.

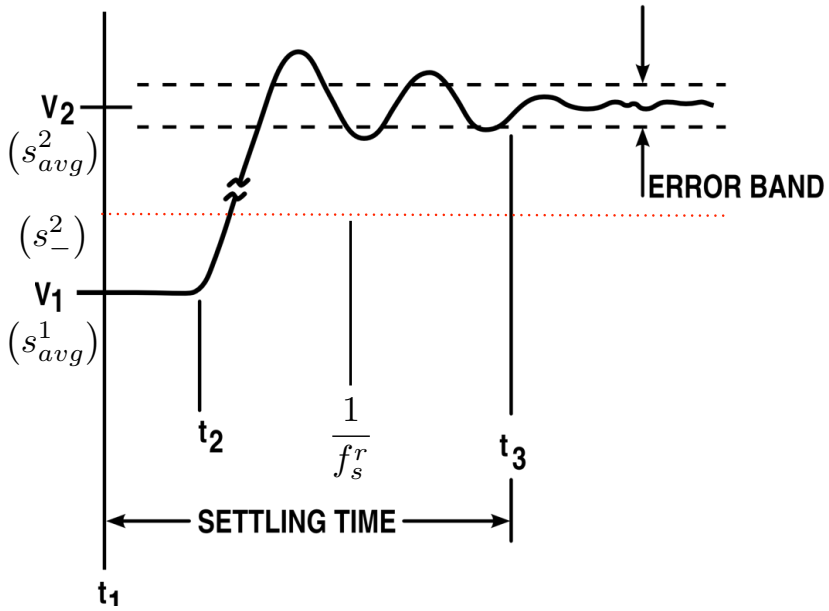

Figure 5.3 Calculating minimum settling time: at time t_1 attacker DAC is alerted to change output from $s_{avg}[1] = V_1$ to $s_{avg}[2] = V_2$; DAC output begins to change at t_2 and achieves steady state by t_3; at $t = 1/f^r_s$, the inverse of the PLI system's sampler, the output of the DAC must be at least $s_-[2]$ (red) to guarantee acceptance by the system (Source: [63]).

Allowing τ_r to equal the slew time plus the recovery time and τ_d the dead time, the minimum settling time for an attacker's AWG is $\tau_s = \tau_d + \tau_r$, where $\tau_d < 1/f_s$ and τ_r is given by

$$\tau_r = \frac{1/f_s - \tau_d}{\max\left(|s_-[j]/s_{avg}[j]|\right)} \tag{5.12}$$

for $j = 1, \ldots, l$.

Signal-to-noise ratio (SNR)

SNR specifies the power of the generated signal to the total amount of noise produced by the DAC [61]. Distortion is excluded from the noise measurement by ignoring a specified number of harmonics from the generated signal. Allowing s to be the signal to be generated by the DAC, s_{gen} the actual signal generated signal (minus distortion), the noise is then $N^s = s - s_{gen}$. Let $P(\cdot)$ be the measure of the power, the SNR is then

$$\text{SNR} = \frac{P\left(s\right)}{P\left(N^s\right)} \tag{5.13}$$

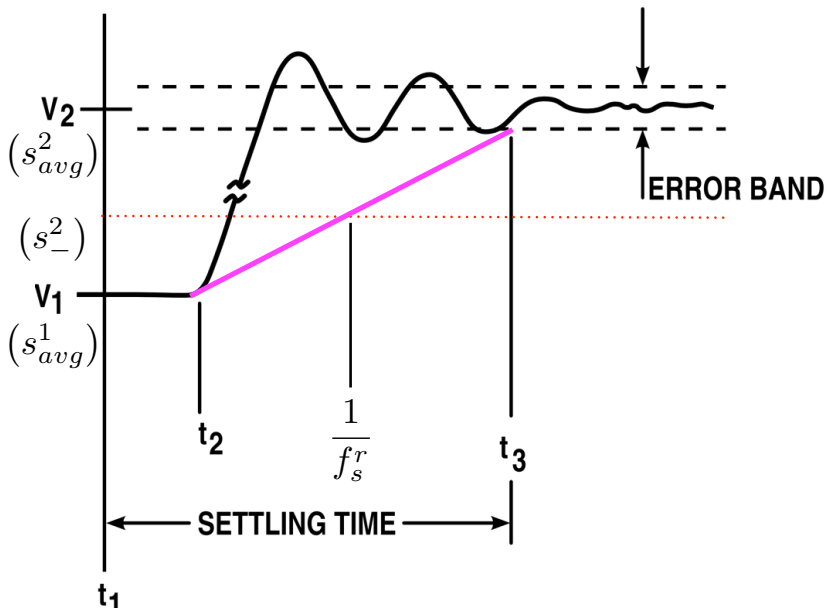

Figure 5.4 Linearisation of slew and recovery times to calculate settling time. Output of DAC is guaranteed to always be greater than purple line.

By definition s_{avg} is free of noise and distortion as it is the signal the attacker is trying to reproduce. To determine the maximum amount of noise an attacker's AWG may generate and still produce a signal within $th_{+/-}$, it is then necessary to add noise to s_{avg} until one or the other of the thresholds is exceeded.

Allowing N to be of a vector of Gaussian noise of length l, the power of N should be increased until

$$\sum_{k=1}^{l} f_g[k] \times (s_{avg}[k] + N[k]) \begin{cases} > th_+ \\ \text{or} \\ < th_- \end{cases} \tag{5.14}$$

The minimum SNR an attacker need maintain is then $P(s_{avg})/P(N)$.

Distortion

Three distortion measurements are commonly used to characterise the performance of a DAC: total harmonic distortion (THD), total harmonic distortion plus noise (THD+N), and signal-to-noise plus distortion ratio (SINAD) [61]. For a DAC the THD is a ratio of the power of the first n harmonics of a generated sine-wave signal to the power of the signal itself. Allowing

H_i^s to denote the i^{th} harmonic of the signal s, the THD is given by

$$\text{THD} = \frac{\sum_{i=1}^{n} P\left(H_i^s\right)}{P\left(H_0^s\right)} \tag{5.15}$$

where H_0^s is the fundamental frequency of s and $P(\cdot)$ is as above. According to [64] the first nine harmonics should be used; i.e. $n = 9$.

For the THD+N calculation a similar ratio of the harmonics-to-signal power is taken but with the total power of the noise of the generated signal, N^s, added to the harmonic noise (the bandwidth over which the noise is measured must be specified)

$$\text{THD+N} = \frac{\sum_{i=1}^{n} P\left(H_i^s\right) + P\left(N^s\right)}{P\left(H_0^s\right)} \tag{5.16}$$

SINAD then is the ratio of signal, noise, and distortion powers to the noise and distortion powers (over a given bandwidth SINAD is equivalent to THD+N)

$$\text{SINAD} = \frac{P\left(H_0^s\right) + \sum_{i=1}^{n} P\left(H_i^s\right) + P\left(N^s\right)}{\sum_{i=1}^{n} P\left(H_i^s\right) + P\left(N^s\right)} \tag{5.17}$$

To calculate the maximum distortion allowed by the PLIS, we must first define what constitutes distortion of the signal s_{avg}. Linear distortion of this signal would be of the form $\kappa \times s_{avg}$, where κ represents an arbitrary constant [52]. As the attacker may deviate from each sample point of s_{avg} by as much as $A_{+/-}$, we are dealing with non-linear distortion; specifically amplitude distortion. The distortion of the signal is then simply the difference between s_{avg} and the attacker's signal, or $A[i]$ as defined in Section 5.3.2. The maximum amount of distortion, $D_{+/-}$, the signal s_{avg} may experience and still be accepted as genuine is then simply the difference between the signals, $s_{+/-}$, that produce filter outputs equal to $th_{+/-}$ and s_{avg}, that is

$$D_{+/-} = s_{+/-} - s_{avg} \tag{5.18}$$

To relate this to our distortion parameters we note that an attacker is actually attempting to generate a signal made-up of a combination of sine waves of varying amplitudes and phases. We must therefore calculate the allowable distortion with respect to each of the individual frequency components of the signal. Allowing $\mathcal{S} = FFT\left\{s_{avg}\right\}$ and $\mathcal{D}_{+/-} = FFT\left\{D_{+/-}\right\}$, the maximum allowable THD of the attacker's AWG for each frequency bin i within the bandwidth

B of the (sampled) signal s_{avg}, is given by

$$\text{THD}[i] = \min\left(\frac{\sum_{j=1}^{9} P\left(\mathcal{D}_{+}[i \times j]\right)}{P\left(\mathcal{S}[i]\right)}, \frac{\sum_{j=1}^{9} P\left(\mathcal{D}_{-}[i \times j]\right)}{P\left(\mathcal{S}[i]\right)}\right) \tag{5.19}$$

Calculating the THD+N and SINAD are not so simple as adding noise to the signal in a manner similar to that used in the SNR calculation outlined above, for it cannot be known the proportion of noise and distortion to use to bring the signal above or below the thresholds as each may affect the PLIS differently (i.e. the PLIS may be more or less resistant to distortion than noise, or vice-versa). An estimate of the THD+N and SINAD must then be specified according to both the level of distortion and noise.

For simplicity's sake, let us consider only the case of D_{+}, that is the distortion necessary to produce a filter output equal to th_{+}, as the same procedure is followed for D_{-} (the THD+N and SINAD being the maximum result of the two). Allowing $D = \kappa D_{+}$, where κ is a scalar between $[0 : 1]$, to denote the amount of distortion applied to the signal s_{avg}, and $\mathcal{D}$ and $\mathcal{S}$ their Fourier transforms, respectively. To calculate the THD+N and SINAD for the i^{th} bin over the bandwidth B, combine each bin of $\mathcal{S}$, except for the i^{th} bin, with the bins of $\mathcal{D}$ that constitute the harmonics of the i^{th} bin, and take the inverse Fourier transform of the resulting signal. Using the procedure set forth above for SNR, add noise to this signal until $th_{+/-}$ is crossed. The power of this noise is used in (5.16–5.17) with the distortion portions of the equations calculated using the combined signal. This procedure is followed for as many levels of distortion (i.e. values of κ) as are deemed necessary.

Spurious Free Dynamic Range (SFDR)

SFDR is a narrow-band measure of distortion/noise in that it is the ratio of the power of the generated signal to the largest spurious frequency component produced by the DAC during generation of the signal [61].

Because the signal the attacker is attempting to reproduce is composed of many frequencies, we must consider a spurious signal for each. Since SFDR is given as the worst case across the AWG's bandwidth, we will assume that recreating a signal with n frequency components creates n spurious signals with a constant SFDR (i.e. the ratio between the frequency component

generated and the spurious signal is the same for each component).

In analysing the affects of SFDR on an attackers ability to forge s_{avg} we must determine how great the SFDR for each of the frequency components used to generate the signal need be for the signal to fall outside of the thresholds for the genuine signal. How these spurious signals are distributed among the bandwidth of the signal has a bearing on the magnitude of their affect. Are they for instance randomly distributed or are they a multiple (harmonic) of the generated frequency component? Our analysis must then consider the number of frequency bins used in recreating s_{avg}, where the spurious components are placed, and how strong they need to be for the generated signal to fall outside $th_{+/-}$.

Allowing $\mathbb{S}$ be an ordered list, from greatest to least power, of the frequency bins of the signal s_{avg} and m the number of bins over the bandwidth of interest, B, select the first n bins from $\mathbb{S}$, for $n = 1, \ldots, m$, to create the frequency content of the attacker's signal, $\mathcal{S}_a$. Generate complex noise, N, for each of these components such that the ratio $N[i]/\mathbb{S}[i]$ is the same for $i = 1, \ldots, n$. This noise should then be distributed randomly across the bandwidth of $\mathcal{S}_a$. The inverse Fourier transform of $\mathcal{S}_a$ is then taken to determine the attacker's time-domain signal, s_a. The SFDR is found by increasing the ratio $N[i]/\mathbb{S}[i]$ until the resulting s_a produces a filter output outside the thresholds of the genuine device. Each time the number of components used to create is $\mathcal{S}_a$ increased, the $N[i]/\mathbb{S}[i]$ ratio should begin at zero to ensure that the number of components used to recreate s_{avg} produces a signal that results in a filter output within $th_{+/-}$ (if not, the resulting SFDR would be $-\infty$). This procedure could also be used to place the noise on arbitrary harmonics of the generated signal to see which configuration results in the largest SFDR. This then is the SFDR an attacker's AWG must meet.

5.3.4 Results for Type I attack

Using the approaches set out above for each of the parameters of interest, we performed an analysis using the device m5c1 from dataset2 (both randomly selected) to determine the characteristics of an AWG necessary to successfully carry out a type one attack on the matched filter methodology. Records 1001–1026 were used to ensure that the device was operating outside the warming-up period noticeable in each device's filter output. A matched filter was

Table 5.1 AWG Characteristics for Type I Attack.

Parameter	Value
Resolution	12 bits
Settling time	0.40025 ns (2498465616 Hz)
SNR	28.025 dB
THD	-75.279 dBc
THD+N	-24.867 dBc
SFDR	14.727 dB

created using sample points 2925–1794 from record 1001 (the same span used for dataset2/3 in Section 4.4.1), with the remaining records used to calculate thresholds. The attacker's signal, s_a, was found by aligning records 1002–1026 to the reference signal and then averaging. A summary of the resulting AWG characteristics is given in Table 5.1; the code used to arrive at these values may be found in Appendix B. A discussion of the calculations for each parameter follows.

Resolution

For dataset2/3 a two channel oscilloscope with 8 bits of resolution was used to capture the differential signals used in 10Mb Ethernet; however, as mentioned in Section 5.1, we have performed the matched filter operation on the reconstructed waveform; i.e. we have subtracted the data obtained on one channel from the other and used the resulting signal for analysis. As such the data used for the AWG characterisation study should actually be considered 9-bit data: the maximum of the absolute value of any of the binary sample points that make up the waveforms was greater than 127 but less than 255; 8 bits, plus another bit for the sign, are required to represent this data then. The y-scale, or voltage, increment used in the capturing routine was 0.02 volts, which leads to an effective full-scale voltage of -5.12 to +5.10 V (binary values for the sample points range from -256 to 255).

Using the method outlined in Section 5.3.3.1 would require an AWG with 12 bits of resolution and the above full-scale voltage to realise an approximation of the signal s_{avg} (i.e. a signal within the bounds of $s_{+/-}$ [Figure 5.5]). The resolution of the attacker's AWG must be higher than the sampler of the PLIS (9 bits) because of the constraints of the type one attack, which

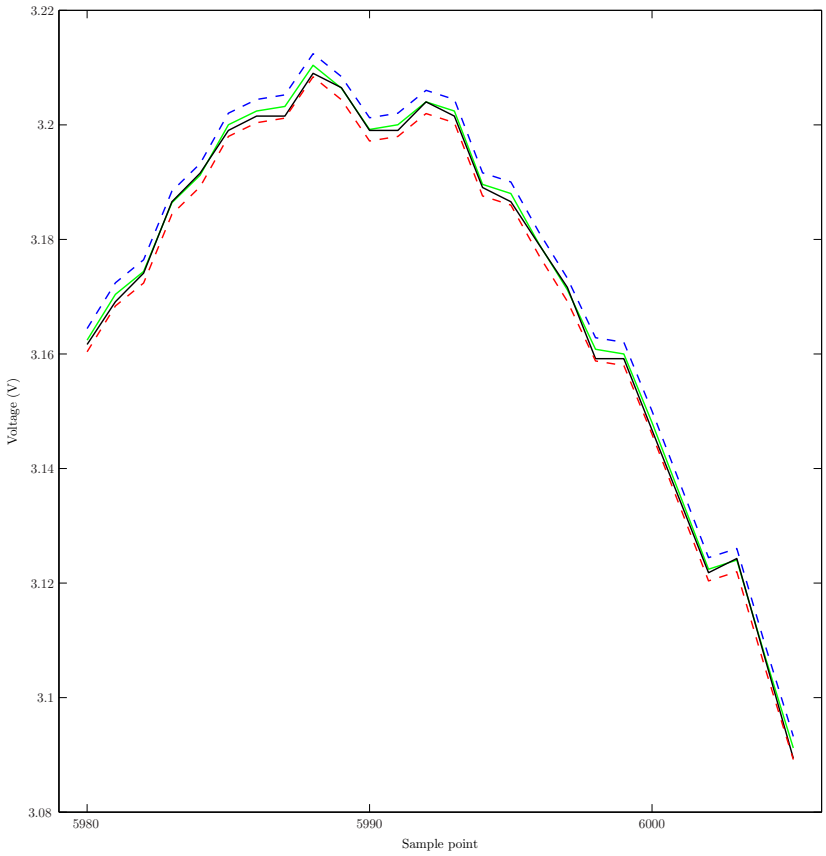

Figure 5.5 Optimal signal to use for attack (green), 12-bit realisation thereof (black), and maximum and minimum values of attack signal (red).

requires that the attacker produce a signal between $s_{+/-}$. Reducing the full-scale voltage of the AWG to better match the minimum and maximum values of the actual waveform (down to say -3.5 to +3.5 V) would decrease the required resolution because, though the difference between s_+ and s_- would not change, the voltage difference between each consecutive binary value would decrease. Another consequence of the model is that increasing the sampling rate, or including more of the Ethernet frame in the filter, reduces the amount of allowable deviation per sample point (i.e., the difference between s_+ and s_- decreases), which in turn necessitates an increase in resolution.

If the restraints of the attack model are lessened so that levels of s_{avg} are instead the nearest multiples of $\pm V_{fs}/(2^n - 1)$, assuming $V_{fs} = 5.10V$, we find that the minimum resolution needed to produce a signal with a filter output between $th_{+/-}$ for the matched filter methodology is

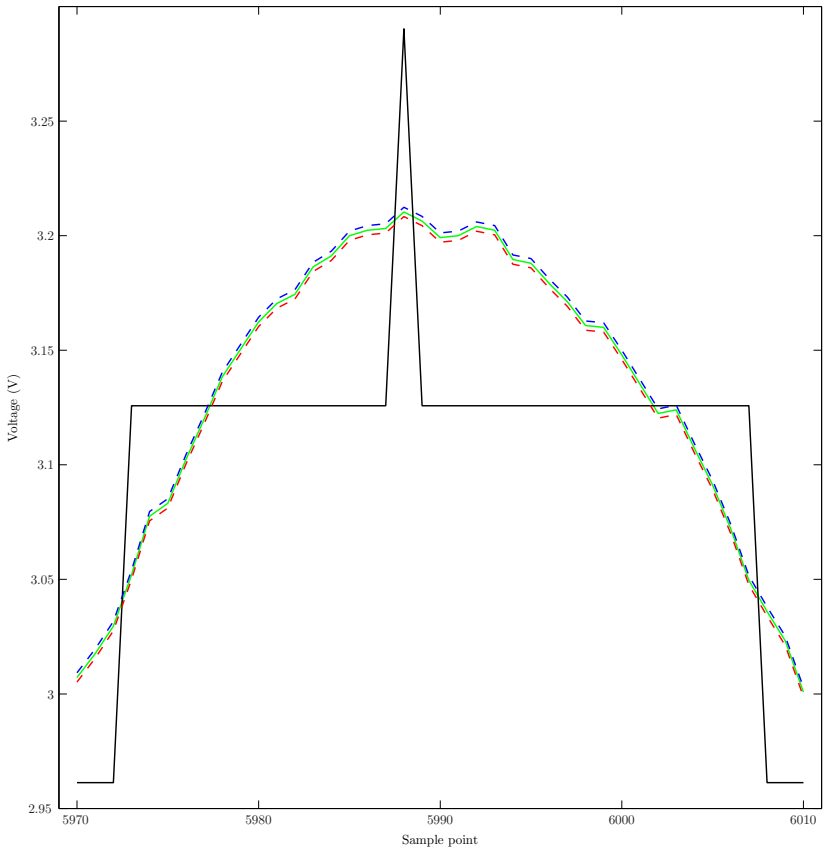

Figure 5.6 Optimal signal to use for attack (green), 5-bit realisation thereof (black), and maximum and minimum values of attack signal (red).

only $n = 5$ bits (Figure 5.6).

In any case, no matter the resolution of the attacker's AWG the final values of the s_{avg} will be determined by the sampler of the PLIS. To ensure that the signal is acceptable to the system it may be necessary to decrease/increase the resolution of the signal to that of the PLIS and modify the resulting levels accordingly.

Settling time

Due to the slight difference between s_- and s_{avg} at the waveform maximum and minimum, the required settling time of the attacker's AWG is very nearly equal to the sampling time of our PLIS (they differ by only 0.0614%).

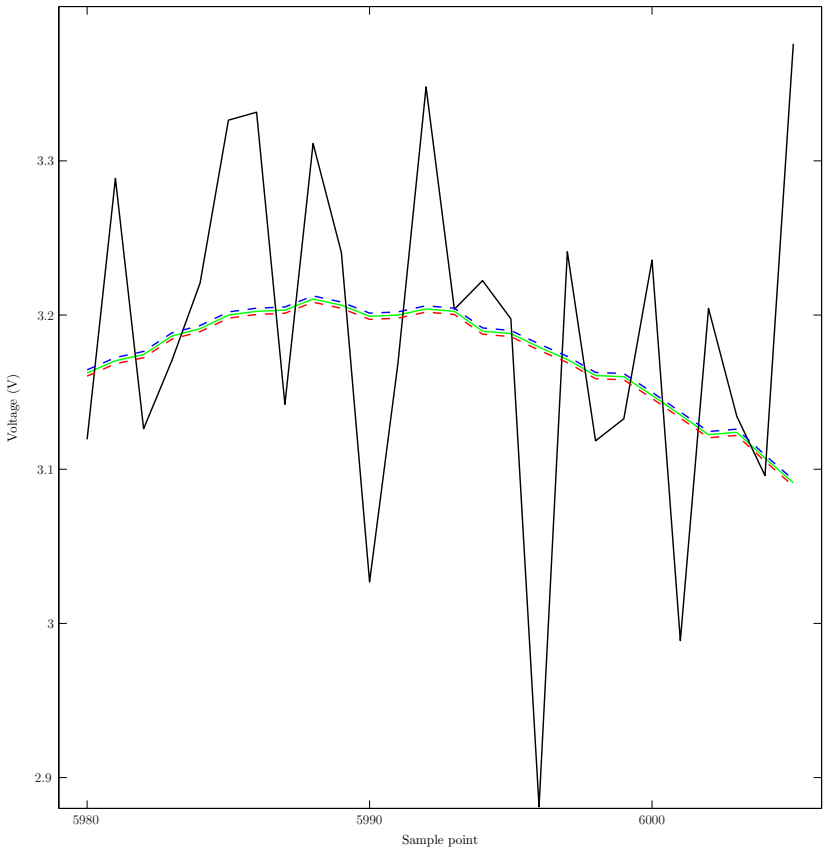

Figure 5.7 Close view of optimal signal to use for attack (green), optimal signal with SNR of ∼28 dB (black), and maximum and minimum values of attack signal (red).

SNR

It was necessary to repeat the procedure given in Section 5.3.3.1 several times (1000 in our calculations) and then average the results to arrive at an accurate estimate for the SNR, as for any individual iteration the additive white gaussian noise generated could decrease or increase the amplitude of particularly consequential or inconsequential (i.e. of greater or lesser amplitude) sample points in a non-uniform fashion.

The maximum estimated SNR necessary to successfully carry out the attack is quite high and results in a signal visibly different from s_{avg} (black lines, Figures 5.7 and 5.8). This is actually to be expected as the matched filter is an optimal detector in the presence of AWGN. This immunity from noise derives from its origins in communications systems, where this behaviour is counted as a strength; in a PLIS must be counted as a deficiency.

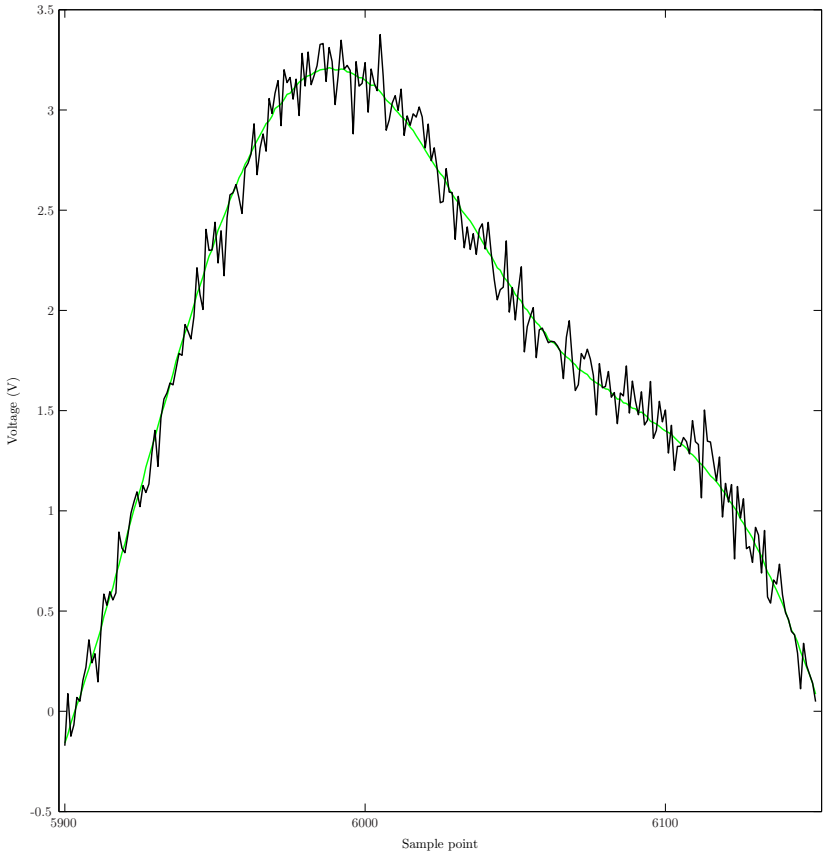

Figure 5.8 View of optimal signal to use for attack (green) with optimal signal with SNR of ∼28 dB (black) to highlight extent of visible differences.

THD and THD+N

Both THD and THD+N calculations were carried out over the bandwidth of 20.013 MHz using nine harmonics, as per IEEE standard 1241-2010 [64]. Figure 5.9 gives the maximum allowable THD per frequency bin to produce s_{avg} within $s_{+/-}$. For an AWG the stated THD is usually taken as the maximum THD across the given bandwidth; thus the THD of the attacker's AWG should be no greater than the minimum THD shown in Figure 5.9. To calculate THD+N ten levels of distortion were used; i.e. $\kappa = 0.1, 0.2, \ldots, 1$. Using the same reasoning as for the THD estimate, the maximum amount of allowable THD+N was determined by taking the minimum value for each distortion level and then averaging. As can be seen from Figure 5.10, which shows the maximum THD+N as a function of distortion level and frequency bin, the estimated THD+N is roughly constant across distortion levels. This merely reinforces the point that the matched filter is insensitive to noise.

SFDR

In our SFDR estimation we assumed that any frequency components used to produce the signal s_{avg} would result in a spurious signal placed at random across the bandwidth of the AWG. The number of components needed to generate the signal under the restrictions of the type one attack are substantially fewer than would be need to be produced in an actual attack, as using too few components would result in misalignment (in our construction of an attack perfect alignment is assumed). As such, the SFDR given is an average of signals produced using n components, where n varied from a single component to every component in the bandwidth of the sampled waveforms used to create s_{avg}. Even then the reported SFDR, which being positive indicates that any spurious signal produced during the generation of an individual frequency component can actually be of greater power than the component itself, results in a signal that is clearly different from s_{avg} and would be distinguishable from a genuine signal using one of the tests discussed in Section 4.2.3 (using the norm test, for instance, decreases the SFDR to -28 dB).

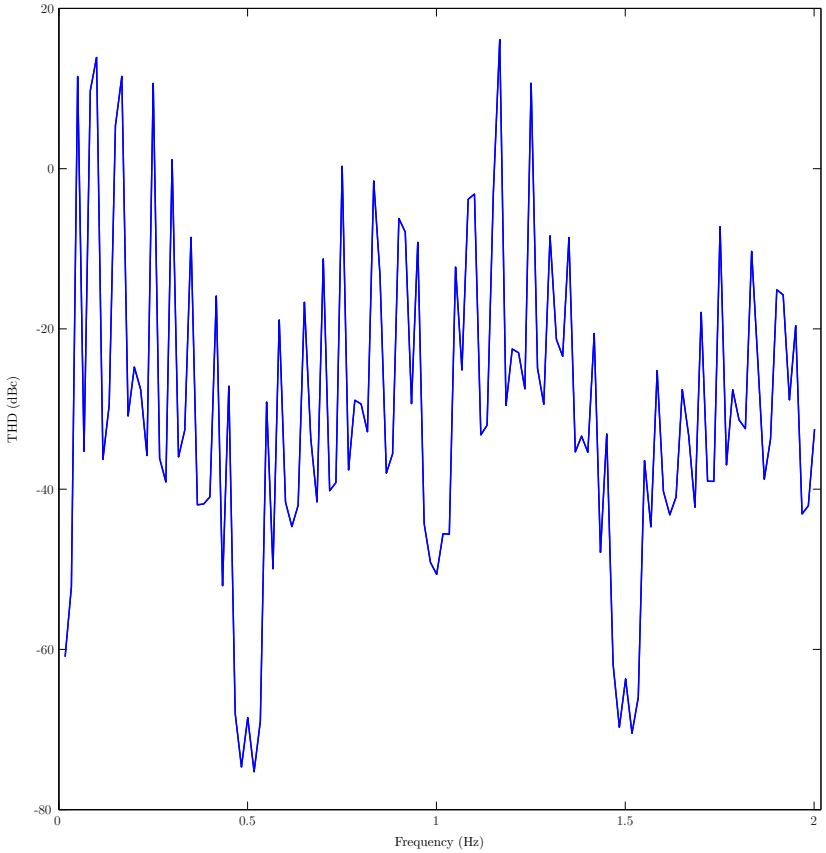

Figure 5.9 Minimum THD, measured with respect to carrier, necessary to produce attack signal guaranteed to be accepted by PLIS.

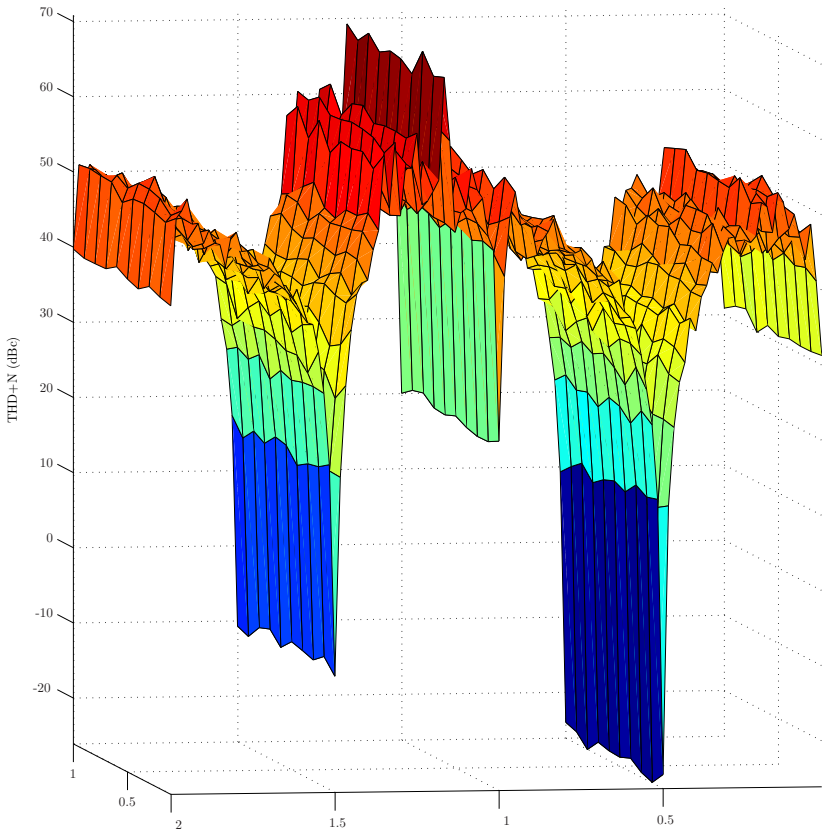

Figure 5.10 Minimum THD+N, measured with respect to carrier (left-to-right, scale is 10MHz)and distortion level (front-to-back, scale is [0:1]), necessary to produce attack signal guaranteed to be accepted by PLIS.

5.4 Type II attack

In a type two attack an attacker is not seeking to produce a high-fidelity copy of a signal from a valid device but rather, in the case of the matched filter, attempts to manipulate the signal generated by their device to produce an output within the thresholds for a device already enrolled in the PLIS system. The only limitation the attacker faces is that the manipulated signal must behave according to the standard; i.e. the voltage levels, signal transitions, etc are in accordance with those specified in [58].

To carry out such an attack, however, requires more knowledge of the PLIS and associated target device than a type one attack. Whereas a type one attack can be carried out simply by observing frames from the targeted device, in a type two attack the attacker must possess both the device's reference signal and thresholds for future outputs to be able to construct their signal. By knowing these, an attacker can manipulate their signal, in whole or in part, to produce a filter output falling within the thresholds for the device.

While the character of the manipulation is specific to the PLIS and its underlying classification technique, for the matched filter methodology, knowing that it is a sum-of-squares operation and as a simple demonstration, an attacker would simply need to amplify or attenuate their signal, s_a, to such a degree as to satisfy

$$th_- \leq \sum_{k=1}^{l} f_t[k] \times (s_a[k] \times A) \leq th_+ \tag{5.20}$$

where f_t is the reference signal for the targeted device, th_+/th_- the thresholds for its filter outputs for the next m records, and A is the amount of amplification/attenuation applied to the signal.

As a practical matter, which follows the reasoning set forth in Section 5.3.1, an attacker would seek to produce a filter output of $(th_+ - th_-)/2$. The amount of amplification/attenuation needed to satisfy (5.20) is then given by

$$A = \frac{(th_+ - th_-)/2}{f_t \cdot s_a} \tag{5.21}$$

As no amplifier or attenuator can be made to have a flat response over all frequencies, an analysis similar to that carried out in Section 5.3.3, involving the maximum amount of

allowable deviation of the constant A for each sample point, say, would need to be carried out to determine the type of technology an attacker need employ to perpetrate the attack.

5.5 Conclusion

We have delineated, and described the information necessary to carry out, two types of attacks against PLIS. For the first type of attack, in addition to having proved the optimal signal for the attacker to construct, we provided the parameters needed to define the type of hardware necessary to produce this signal and a methodology for determining the values of those parameters. These parameters characterise the performance of an AWG needed to successfully defeat a PLIS, and can be taken as a sort of metric indicating the security of a given PLIS: if the parameters indicate that very high-end equipment is required then the number of attackers will be constrained by economics; it may even be possible that no such equipment that satisfies the parameters exists at the present time, in which case it may be concluded that the PLIS under examination is at least temporarily secure against the type one attack.

When designing a PLIS system, then, the classification technique should be chosen so as to be as sensitive as possible to these parameters, so that it may be resistant to forged signals, as well as its ability to differentiate between a population of devices. It is also possible,though, that while a classifier is theoretically sensitive enough to distinguish between an actual signal and the most high fidelity forgery, all Ethernet devices may exhibit an inherent amount of variation in their signals, from frame-to-frame, that is greater than the minimal difference that can be produced between authentic and forged signals. Such an occurrence would render a high sensitivity to the parameters laid out above inconsequential.

Having applied the methodology to the matched filter, we find that the threat model is biased too much towards the attacker and that the analytical portion of the methodology should be replaced by simulations. The main deficiency of the model is that optimum alignment between the attacker's signal and the matched filter is assumed. This makes the PLIS appear much weaker than it actually is, as heavily distorted and noisy signals can be created, given *a priori* alignment, that satisfy the thresholds but would actually align properly.

The boundaries for the maximum amount of deviation per sample point discussed in Section 5.3.2 are too strict, in that signals having an equal number of sample points of the same magnitude above and below these boundaries, for example, would be accepted by a PLIS, as illustrated by our discussion of resolution in Section 5.3.4. A more accurate estimation of the parameters could be obtained by first selecting a realisable s_a—e.g. having calculated s_{avg} determine the minimum bit resolution for it to be accepted by the PLIS—and then performing simulations using this signal to determine the remaining parameters. By replacing an analytical approach with one based mostly on simulation, we would also be able to relax the requirement of perfect alignment. This would, however, require that modelling for the different types of noise and distortion be specified. For example, should the distortion used in THD calculations be random (i.e. the magnitude and signs of the real and imaginary components are randomly generated) or is there a more exact between the harmonics?

CHAPTER 6. POPULATION SENSITIVITY

The population sensitivity of a classifier sets a theoretical upper bound on the number of devices it is capable of distinguishing between. Should that bound be substantially lower than the total device population, the classifier would be unsuitable for deployment as collisions between cards would be inevitable. In this chapter we show how the population sensitivity of the matched filter methodology can be estimated using theoretical maximum and minimum filter outputs derived from the 802.3 standard and the measured variance of filter outputs.

6.1 Matched filter population sensitivity

From the discussion of the matched filter methodology in Section 4.2.2, we know that for two devices to be distinguished the thresholds computed for their filter outputs must not overlap (this is illustrated in Figure 4.8). Let us assume the difference, denoted by δ, between the upper and lower thresholds, $th^i_{+/-}$, for all $i = 1 \ldots n$ devices are identical. This is not to say that $th^i_{+/-}$ are the same for every devices but only that the difference, $th^i_+ - th^i_-$, is. (Which is to say, in reference to Equation 4.13, the filter outputs for each device, while having a different mean, would share the same standard deviation.) Furthermore, assume that c_+ and c_- are, respectively, the maximum and minimum filter outputs possible. The number of devices we would be able to distinguish would then depend on how many times the difference between c_+ and c_- was divisible by δ, i.e.

$$\text{number of devices } = \frac{c_+ - c_-}{\delta} \tag{6.1}$$

In graphical terms, we wish to know how many times the space between c_+ and c_- may be partitioned into non-overlapping areas of the width δ (Figure 6.1).

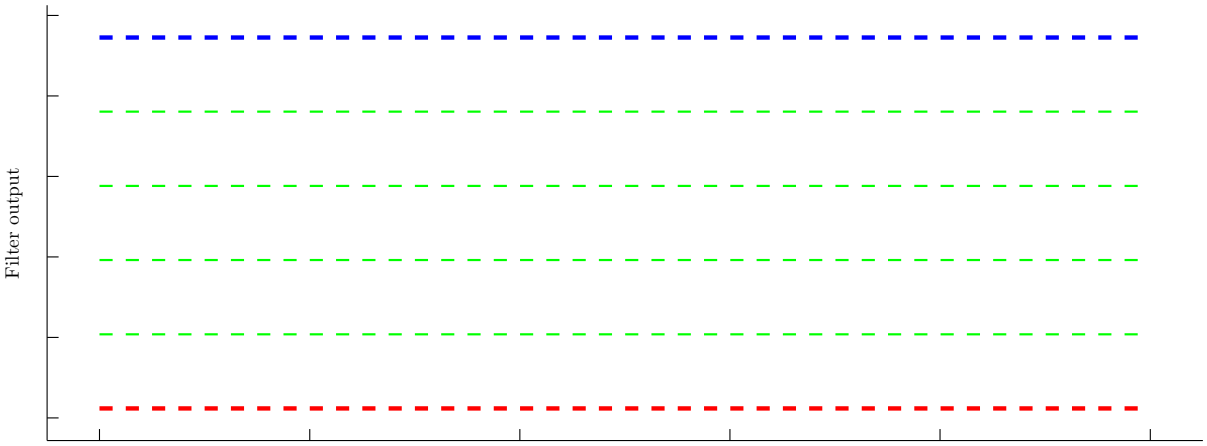

Figure 6.1 Graphical representation of population sensitivity estimation. Partitioning space between c_+ (blue) and c_- (red) by thresholds of width δ (green-to-green) allows the classifier to distinguish up to five devices.

In the determination of $th_{+/-}$ and $c_{+/-}$, we must take into account that the output of the matched filter and the thresholds used to identify devices are functions of the sample frequency and bit resolution of the oscilloscope used to capture signals and the underlying variability of that signal.

The IEEE 802.3 standard [58] sets forth the minimum voltage, V_-, and maximum voltage, V_+, values of the waveform for each bit value, which in Manchester encoding consists of a transition or lack thereof (Figure 6.2). Thus a continuous-time representation of the upper (lower) limits of, say, the synchronisation signal can be constructed from V_+ (V_-) by appending copies of V_+ (V_-), or its inversion, to signify a lack of transition, to itself in the correct pattern to produce the 56-bit sequence of alternating ones and zeros, followed by `10101011`, that makes up the signal.

These synthesized continuous-time synchronisation signals need to be made discrete in voltage and time according to the resolution and sampling frequency of the oscilloscope used in the PLI system. Allowing s_+ and s_- to denote the maximum and minimum discretized versions of the synchronization signals, we then construct matched filters for the upper and lower voltage ranges. If sp_f and sp_l give the first and last sample points of the span of the synchronisation signal used to create these matched filters, the maximum and minimum filter outputs possible

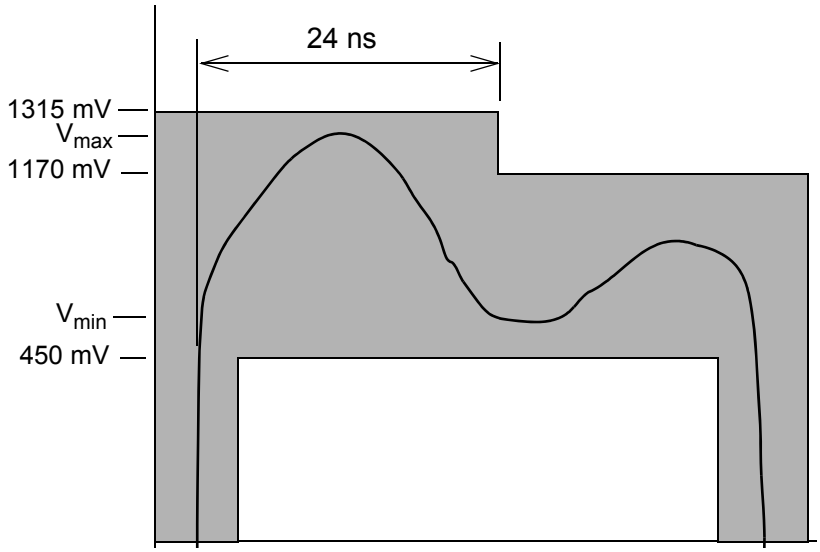

Figure 6.2 Voltage and timing limits of 10Mb Ethernet signal; differential output, half of a bit period (Source: [58]).

are given by $c_+ = s_+[sp_f : sp_l] \cdot s_+[sp_f : sp_l]$ and $c_- = s_-[sp_f : sp_l] \cdot s_-[sp_f : sp_l]$, respectively.

For the second part of (6.1), the difference between the thresholds, δ, is calculated by first creating matched filters for each of the devices in our dataset(s) using sample points sp_f through sp_l of their reference records (these may need to be adjusted slightly on a per-device basis to correspond to the voltage levels given by $s_+[sp_f : sp_l]$ and $s_-[sp_f : sp_l]$). Having applied these filters to each device to determine its control response, we take the average of the difference between thresholds calculated for each 20 record interval (which are found by following the procedure set forth in Section 4.3.3) for each device. By selecting the smallest and largest average difference, δ_{max} and δ_{min}, respectively, from all the devices, we are able to establish a range for the number of cards the matched filter is capable of distinguishing between

$$\frac{c_+ - c_-}{\delta_{max}} \leq \text{ number of devices } \leq \frac{c_+ - c_-}{\delta_{min}} \tag{6.2}$$

We should note that the 802.3 standard actually sets limits (pictured above) on the differential signal at the output (secondary side of transformer) of the transmitter but that our experimental setup currently measures the signal on the secondary side of the receiver's transformer. Therefore, the records in our datasets would need to scaled appropriately (amplification or attenuation of the signal would depend on the ratio of the transformer windings at the re-

ceiver) before δ could be calculated.

6.2 Conclusion

We have outlined a method by which it is possible to estimate an upper bound for the number of devices the matched filter methodology, or indeed any PLIS making use of two-sided thresholds and operating on a signal(s) with constraints set on its electrical behaviour, is capable of distinguishing between. We say that we have provided an upper bound for two reasons. The first being that devices in practice devices may operate within bounds much narrower than the standard prescribes or perhaps only a multitude of sub-ranges. Secondly, our method implicitly assumes that any signal within s_+ and s_- is realisable; this allows for signals of arbitrary bandwidth and resolution.

The first of these points could be investigated empirically by studying a statistically significant quantity of devices—of different models, date of manufacture, and under varying conditions—to determine how manufactured devices behave. The second requires that only signals within a limited bandwidth and resolution be considered (i.e. the intervening signals should sampled at the PLI system's sampling frequency and resolution). Having calculated the number signals possible within PLI system's bandwidth and resolution, (6.2) could be scaled appropriately.

CHAPTER 7. ORIGIN OF VARIATION

In this chapter we set forth a methodology that allows one to determine whether or not a particular device component causes, or contributes significantly to, the differences in signalling behaviour between cards that allow for their identification.

7.1 Modelling components

When we speak of modelling a component or components of a device, we mean that we wish to determine what the output—either the voltage, current, or both—of said component(s) will be given a certain input voltage or current. Component behaviour may be captured in one of two ways: using a lumped, or discrete, circuit model or by viewing the component of as a kind of black box, wherein an input simply produces an output without explanation.

A discrete model consists of resistors, inductors, and capacitors arranged in such a way so as to mimic the response of the component to an input. Each of these discrete elements accounts for how energy is accounted for in the component; electrical energy is represented by capacitance, magnetic energy by inductors, and dissipated power by resistors [65]. Having constructed such a model, measurements of the component being modelled must be carried out in such a way as to reveal the values of elements used in the model. These types of models are advantageous in that not only can they provide closed form solutions for the output, but they also allow us to think about the inner-workings of the component using well understood processes (i.e. how the different circuit elements used to model the component interact, etc).

The black box, or port, model of a component specifies only the port characteristics of the device—i.e. it only indicates what the voltage/current will be at one port given a voltage/current applied at another—but not why this is so. While this model may not explain the

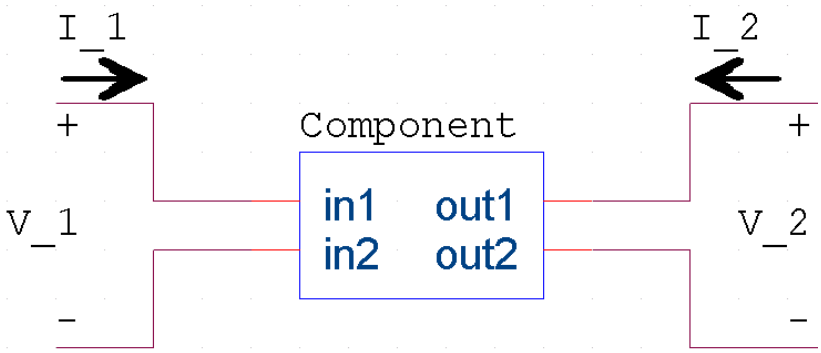

Figure 7.1 Depiction of a two-port model for a component (input voltage/current denoted by V_1/I_1 and output voltage/current by V_2/I_2). Note: it is assumed that voltage/current measurements are carried out at device terminals.

behaviour of a component, it does capture the behaviour of a component precisely, within the limits of the measured inputs/outputs and under the assumption of linearity.

In a two-port model (Figure 7.1) an input voltage, V_1, and current, I_1 are related to the output voltage, V_2, and current, I_2 via linear combination. Given four variables, there are six ways to choose two dependent and two independent variables; these six choices represent the possible two-port models (also called parameters) we must choose from. (It is actually slightly more complicated than this as we must choose which dependent variable is be written first. Furthermore, linear combinations of independent variables with linear combinations of dependent variables further increases the number of possible equations to infinity. Refer to [65] for a detailed discussion.) The parameter type chosen usually depends on how multiple two-port models are to be connected [66].

Before proceeding with our discussion of how a two-port model of a component can be constructed in order to determine the influence the component has on the unique behaviour of the device, it should be noted that if the operating frequency of the component being modelled is is high, as it is in networking technologies beyond 10Mb Ethernet, then it may be necessary to use two-port models where the independent/dependent variables are themselves linear combinations of independent/dependent variables (see Section 7.2). In addition, a two-port model of a device is only valid if a true ground plane exists; i.e. the ground is of zero potential, zero resistance, and is continuous [65].

7.1.1 ABCD parameters

For our analysis we chose ABCD parameters because of the ease of combining ABCD models for multiple components in a cascaded or chained fashion (when using ABCD parameters with multiple components connected in a cascaded configuration, it is only necessary to perform simple matrix multiplication to determine their combined response). This allows us to build more complicated models, in which additional component models are added to the chain, to see how different components affect a device's signal in combination with ease.

ABCD parameters treat V_1 and I_1 as dependent variables and V_2 and I_2 as independent ones; the input voltage/current and output voltage/current are related via

$$\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} V_2 \\ -I_2 \end{bmatrix} \tag{7.1}$$

where each of A, B, C, and D are defined as [66]

$$A = \left.\frac{V_1}{V_2}\right|_{I_2=0} \tag{7.2a}$$

$$B = \left.\frac{V_1}{-I_2}\right|_{V_2=0} \tag{7.2b}$$

$$C = \left.\frac{I_1}{V_2}\right|_{I_2=0} \tag{7.2c}$$

$$D = \left.\frac{I_1}{-I_2}\right|_{V_2=0} \tag{7.2d}$$

In characterising a device that operates over a bandwidth, it is necessary to determine ABCD parameters at multiple frequencies. The question of how ABCD parameters for a component may be obtained is discussed in Section 7.2.

7.1.2 Proposed model

To determine the affect of an arbitrary device component on the voltage signal V_s, we propose the following model where Z_S represents the impedance of the source generating V_S, Z_L is an arbitrary load with ABCD parameters of $\left[\begin{smallmatrix} 1 & 0 \\ 1/Z_L & 1 \end{smallmatrix}\right]$, and the box M represents the ABCD parameters, denoted by $\left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$, at the frequency of V_S for the component under consideration. The voltage across Z_L is then the modified signal V_S, which is found by solving

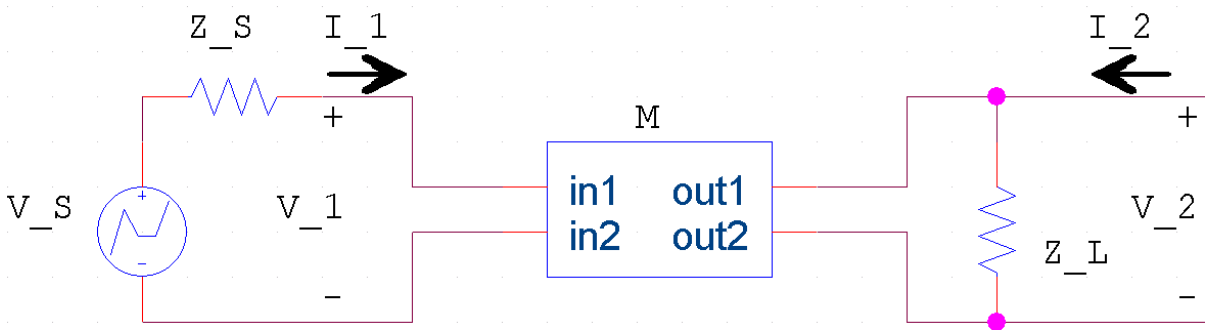

Figure 7.2 Model to examine how an input signal (V_S) is affected by a component with ABCD parameters of M (Z_S is the impedance of the source generating V_S and Z_L is the impedance of a test load).

the following set of equations

$$\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 1 & 0 \\ Y_L & 1 \end{bmatrix} \begin{bmatrix} V_2 \\ -I_2 \end{bmatrix} \tag{7.3a}$$

$$= \begin{bmatrix} a + bY_L & b \\ c + dY_L & d \end{bmatrix} \begin{bmatrix} V_2 \\ -I_2 \end{bmatrix} \tag{7.3b}$$

where $Y_L = 1/Z_L$. As the output port is an open circuit, $I_2 = 0$; furthermore, we need only consider the output voltage, V_2, so (7.3b) reduces to a single equation, which after rearranging

$$V_2 = \frac{V_1}{a + bY_L} \tag{7.4}$$

The input voltage, V_1, may be found by means of a voltage divider

$$V_1 = \frac{Z_{IN}}{Z_{IN} + Z_S} V_S \tag{7.5}$$

where Z_{IN} is the input impedance of the cascade (the component plus the shunt load). As per the definition of ABCD parameters, dividing (7.2a) by (7.2c) gives $Z_{IN} = A/C$. The input impedance needed for (7.5) is then

$$Z_{IN} = \frac{a + bY_L}{c + dY_L} \tag{7.6}$$

Allowing the source impedance to equal the load impedance, $Z_S = Z_L$, substituting (7.6) into (7.5), and then using the result with (7.4) leads to the following expression for the output

voltage

$$V_2 = \frac{V_S}{a + d + bY_L + c/Y_L} \tag{7.7}$$

Equation 7.7 thus describes how the signal V_s would be affected by passing through an arbitrary component with measured ABCD parameters of $\left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$.

7.2 Measuring parameters

As the operating frequency of a component increases, the wave-nature of input/output signals is manifested; i.e. wires are better thought of as transmission lines and signals take on the form of travelling waves. For this reason high-speed devices are often characterised using scattering parameters (S-parameters), which can be defined in terms of travelling waves, instead of ABCD parameters. As scattering parameters consist of linear combinations of voltages and currents at the input and output terminals of a component—again, depending on whether or not a measurements can be taken directly at the terminals, these may or may not be defined in terms of travelling waves—and are thus convertible to ABCD parameters.

For illustrative purposes, we shall assume that voltage and current are measured at the terminals of the component; i.e. wave phenomena may be ignored. While this simplification is acceptable for 10Mb devices (attenuation on the transmit pins of a 10/100 Ethernet card is at least 5 dB at 20MHz and 35 dB at 40MHz [67]), high-speed networking devices would require S-parameters defined in terms of travelling waves (see chapter three of [65]).

In reference to Figure 7.1, voltage-referenced S-parameters for a two-port component are defined as [65]

$$S_{11} = \left.\frac{V_1 - I_1 Z_1^*}{V_1 + I_1 Z_1}\right|_{V_2 = -I_2 Z_2} \tag{7.8a}$$

$$S_{21} = \left.\frac{V_2 - I_2 Z_2^*}{V_1 + I_1 Z_1}\sqrt{\frac{|Re\,(Z_1)|}{|Re\,(Z_2)|}}\right|_{V_2 = -I_2 Z_2} \tag{7.8b}$$

$$S_{12} = \left.\frac{V_1 - I_1 Z_1^*}{V_2 + I_2 Z_2}\sqrt{\frac{|Re\,(Z_2)|}{|Re\,(Z_1)|}}\right|_{V_1 = -I_1 Z_1} \tag{7.8c}$$

$$S_{22} = \left.\frac{V_2 - I_2 Z_2^*}{V_2 + I_2 Z_2}\right|_{V_1 = -I_1 Z_1} \tag{7.8d}$$

where Z_1 and Z_2 are the impedances of the source and load, respectively, causing the excitation of the component. Under the assumption that $Z_0 = Z_1 = Z_2$ and that the source and load impedances used in Figure 7.2 are equivalent to Z_1 and Z_2, respectively, the equivalent ABCD parameters are [65]

$$A = \frac{(1 + S_{11})(1 - S_{22}) + S_{12}S_{21}}{2S_{21}} \tag{7.9a}$$

$$B = \frac{(1 + S_{11})(1 + S_{22}) - S_{12}S_{21}}{2S_{21}} Z_0 \tag{7.9b}$$

$$C = \frac{(1 - S_{11})(1 - S_{22}) - S_{12}S_{21}}{2S_{21}} \frac{1}{Z_0} \tag{7.9c}$$

$$D = \frac{(1 - S_{11})(1 + S_{22}) + S_{12}S_{21}}{2S_{21}} \tag{7.9d}$$

S-parameters can be measured using an oscilloscope and a phase meter; however, this approach becomes tedious when it is necessary to measure the parameters across a range of frequencies. Thus, network analysers are more commonly employed to determine how a component will respond to an arbitrary input, within a given bandwidth. Both vector and scalar network analysers exist; the former provides complex S-parameters, which tell how the magnitude and phase of a signal would be affected by the component, while the latter gives information only about how the magnitude of a signal would be altered. For the matched filter methodology, both the magnitude and phase of the signal are important for device identification, as, while the phase of the fundamental frequency is irrelevant (the filter will be aligned to it), if the phase of the harmonics is modified destructive interference may result. Similarly, if the magnitude of any of the frequency bins is affected the signal amplitude may be attenuated or amplified, depending on the character of the component, either of which might produce a signal with a filter output outside of the thresholds set for the device.

7.3 Determining component significance

In order to discover whether a particular component plays a significant role in device variation, models of the component drawn from several Ethernet cards, based on their measured S-parameters, would need to be constructed. An idealised signal, derived by averaging the waveforms from each of the devices, for each model of card, would serve as the input to the

model. Matched filters for each device would be built using the output of the model and applied to each devices output in turn. If signal variation originates at the component, we would expect to see differences in filter outputs, though even stronger evidence of this affect would come in the form of collisions between the simulated signals from devices known to have shown significant overlap from the experiments. The extent to which a particular component contributes to the uniqueness of a device could be estimated using a quantity of information calculation.

7.3.1 Constructing model input

As the cards studied in Section 4.4 were easily differentiable across model type, an idealised signal (i.e. a signal composed of attributes unique only to the model of the cards), A_i, for each of the i models of Ethernet cards could be constructed by taking n records, selected at random, from each of the device's datasets, aligning, and then averaging. This signal, A_i serves as the input, V_S, to the model depicted in Figure 7.2.

7.3.2 Producing model output

The output of the model constructed for the j^{th} device, V_2^j, is found by using (7.7) with $V_S = A_i$

$$V_2^j = \frac{A_i}{a_j + d_j + b_j Y_L + c_j / Y_L} \tag{7.10}$$

where device j is a member of model i and a_j, b_j, c_j, d_j are the ABCD parameters derived from the S-parameter measurements, for the j^{th} device, of the component being tested. The value of Z_S, and hence Z_L, according to the assumptions laid out above, should be equal to the source impedance used in the S-parameter measurements.

It must be remembered that ABCD parameters are defined as a function of frequency. Thus, it is necessary that ABCD parameters be found for each frequency bin of A_i and then be applied only to the corresponding bin. More explicitly, allowing $\mathcal{A} = \mathscr{F}\{A\}$, the output for the k^{th} bin is

$$\mathcal{V}_2^j(k) = \frac{\mathcal{A}_i(k)}{a_j(k) + d_j(k) + b_j(k) Y_L + c_j(k) / Y_L} \tag{7.11}$$

where $a_j(k), b_j(k), c_j(k), d_j(k)$ are the ABCD parameters measured at the frequency of the k^{th} bin. The time-domain output of the model for the j^{th} device's component is then $V_2^j = \mathscr{F}^{-1}\left\{\mathcal{V}_2^j\right\}$. As each bin is of finite width, it is not possible to have exact ABCD parameters; interpolation between measured parameters, to fill in for unmeasured frequencies, and/or the addition of multiple frequencies, to account for the frequencies contained within a bin, must therefore be used.

7.3.3 Evaluating model output

In seeking to determine the affects a particular component has on a signal, we are actually asking two questions: how much of an affect does the component have (i.e. how significant is the component: is it enough to explain the differences we see between cards?) and does it actually produce the unique characteristics our classifier uses to differentiate devices?

Significance

To calculate the significance a component has in determining device identity, we can calculate how much information is added to the ideal signal by the device model and check whether it is enough to make up the difference between the information of the ideal signal and the actual measured waveforms of the device. Specifically, allowing $u_A = I(A_i)$ to be some measure of the information of A_i, we then measure the average information, u_j, contained in n of device j's records (these records must be aligned to A_i). The ratio

$$\frac{I\left(V_2^j\right) - u_A}{u_j - u_A} \times 100 \tag{7.12}$$

then tells us what percentage of unique information the component is responsible for.

Identity

Even if a component can be shown to add a great deal of information to a signal, we must still find out whether it is information that develops a device's identity. To do this, we generate m test signals, based upon the appropriate A_i, for each device, pass them through the device's model, select one of the modified signals to act as the device's matched filter, and, following

the procedure laid out in Section 4.3, produce filter outputs for each device against itself and all the other devices.

To simulate the natural variability observed in the captured waveforms, a small amount of additive white gaussian noise should be added to each test signal. So long as the variances of the devices' actual filter outputs are approximately equal, this will produce an equivalent overlap between the filter outputs, should any exist. If the filter outputs diverge, we can infer that the component does in fact contribute to a devices identify; significantly, if any collisions occur and happen to correspond with the collisions identified in Tables 4.3–4.5, we can infer an even stronger relationship between the device's identity and the component.

7.4 Conclusion

We have proposed a methodology, based upon an empirical two-port model, capable of determining the extent to which an arbitrary component of a device contributes to its unique behaviour, as manifested by a PLIS. The methodology describes how a component should be measured to create a model that captures component behaviour over the entire bandwidth of its operation, as well as how an input signal that represents a generic device can be used with the model to evaluate the component's affect on creating device identity.

For future work, we suggest that the methods outlined above be applied to the transformer found on all 10/100 Ethernet cards. More specifically, for the devices available to us each transmit/receive pair is connected to an IC that houses a transformer in addition to a low-pass filter (the cutoff frequency and filter order differs between the transmit and receive pins [67]). We believe that the transformer likely contributes to the uniqueness of a device's signal because of the degree to which the transformer and low-pass filter must affect the signal (the transformer increases or decreases the amplitude of the signal while the low-pass filter shapes it).

As this transformer is actually a four-port component our methodology would require some slight modification. The ABCD parameter model would have to be reworked to account for the increase in variables from four to sixteen. Specifically, the derivation for the component's output would need to be altered to reflect the fact that the component produces two outputs

instead of one. As the transformer is employed in the transmission of a differential signal, it should be measured using a setup appropriate for differential four-port S-parameters. If a proper four-port analysis is infeasible, an approximate analysis could be effected by performing two-port measurements, wherein the two ports not being measured are terminated using a load impedance equal to the output impedance of the network analyser, and then combining the resulting S-parameters to form a four-port matrix. The drawback of this simplification is that the influence the disconnected ports have on the measured ports would not be taken into account. Finally, evaluating the model output for a four-port device necessitates that the idealised signal be composed of the unreconstructed differential signals.

CHAPTER 8. FUTURE WORK

Specific avenues of future work for each of the areas discussed in this work have been proposed in the conclusion sections of each chapter: the longterm tracking of devices and forensics applications (Section 4.5); evaluation of the security of PLIS (Section 5.5); determining how many devices a PLIS is capable of distinguishing between (Section 6.2); and where and how device identity originates (Section 7.4).

In what follows we outline specific courses of research to investigate: 1) whether our ability to differentiate between a small sample of devices implies the ability to differentiate the entire device population (extent of variation); and 2) how multiple features of a device's signal can be used in combination to come to a decision regarding its identity for high-speed networking technologies and to thwart type one and two attacks (increasing difference sensitivity).

8.1 Extent of variation

Up until this point, all experimental work (including our own) concerning proposed PLIS have lacked the necessary statistical rigour and proper design to be able to draw conclusions about the efficacy of either the specific approach or PLI in general. Samples of devices used in experiments range from fewer than five to slightly more than 100, and appear to have been selected mainly due to their ready availability. Certainly preliminary experimentation, carried out on a small scale, should be performed before embarking on larger scale experiments. PLI has matured to the point (both in techniques and methodology), however, to warrant a proper investigation into whether or not large numbers of devices can in fact be distinguished. The only way to achieve this is through sample surveying. We therefore propose to perform a, or devise a framework for, proper sample survey of devices to measure the anticipated real world

performance of a PLIS.

Outline of proposed research

The simplest, though not the most widely used, form of sample survey is simple random sample without replacement (SI) [57]. Under SI we assume that the probability of picking the sample s, of size n, from a population U, of size N, is $p(s) = 1/\left(\frac{N}{n}\right)$, while the probability of selecting any particular member k of the population U is $\pi_k = n/N$ and the probability of selecting any two members, k and l ($k \neq l$), is $\pi_{kl} = \frac{n(n-1)}{N(N-1)}$. To determine the mean of some population parameter, denoted by $\hat{\bar{y}}_{U_\pi}$, we select a sample and sum over all members of the sample, dividing by n

$$\hat{\bar{y}}_{U_\pi} = \frac{\sum_s y_k/\pi_k}{N} = \frac{\sum_s y_k}{n} = \bar{y_s}$$

The 95% confidence interval for $\bar{y_s}$ is given by $\bar{y_s} \pm 1.96\left[\hat{V}\left(\bar{y_s}\right)\right]$ where $\hat{V}\left(\bar{y_s}\right)$ is the estimate of the variance of $\bar{y_s}$, given by

$$\hat{V}\left(\bar{y_s}\right) = \frac{1 - n/N}{n}\frac{1}{n-1}\sum_s \left(y_k - \bar{y_s}\right)^2$$

It is important to note that increasing n results in a smaller variance estimate and hence a tighter interval, so it is always better to have as large as sample size as possible.

For our sample survey, estimating the population mean of the APRS metrics set out above would be used to determine the performance of a PLIS in a real world deployment. There are, however, several parameters and design considerations that must be defined before a survey can be carried out. For instance, while the SI surveys are easy to formulate they often require a large number of samples to produce a reasonable estimate (the confidence interval tends to be too wide due to a small n). In practice, auxiliary variables that correlate with the population parameter being measured are used reduce the variance estimate [57]. Thus far, we have failed to correlate filter overlap, in the case of the matched filter, to either distances between MAC address or chipset markings. As the number of elements in the sample, n, will be limited by the number of cards donated/purchased by the researchers or lab computers made available to them, a sufficiently large n may not be forthcoming. The way in which the cards are procured may also have an impact on $p(s)$ (this probability is of equal importance to the attacker) and it

must be determined how large the population N is to be (while 2^{48} MAC address are possible, using this number for N would render a rather wide estimate for the APRS metrics).

8.2 Increasing difference sensitivity

Preliminary work, undertaken using data records that consist of $10,000$ samples points acquired at 1 Gigasamples/s, indicates that the matched filter is unable to distinguish between Ethernet devices operating in 100Mb mode, though this may be due to the hardware used in capturing 100Mb data. As the signalling behaviour of 100Mb Ethernet is dramatically different from 10Mb, it is difficult to acquire accurate samples suitable for use in PLI (i.e. to find a portion of the signal that is recurrent and ubiquitous).

Furthermore, as the matched filter is a sum-of-squares operation it is vulnerable to a simple amplitude modulation attack (Section 5.4). Though this type of attack could be defined as outside any realistic threat model, it nonetheless is at least theoretically possible. We therefore propose to make use of a multi-dimensional change detection framework (ChDF) [68] to increase our ability to distinguish between Ethernet devices operating in 100Mb mode and improve upon weaknesses in the matched filter PLIS.

Outline of proposed research

Allow S to represent a $N \times k$ (observation-by-variable) multi-dimensional data set derived from the distribution $\mathcal{S}$. A generic formulation of the ChDF allows us to answer the question of whether another set of data, $T = M \times k$ $(M < N)$ also originates from $\mathcal{S}$ (the null hypothesis). To do so, we first partition $S = S_1 \cup S_2$, then select M samples at random from S_2 (denote these samples as R), we then calculate whether $d(S_1, R) < d(S_1, T)$, where

$$d(S_1, X) = -\sum_{j=1}^{|X|} \log \sum_{i=1}^{|S_1|} \frac{1}{|S_1|} G(\Sigma_i, X(j) - S_1(i))$$

and Σ_i is the bandwidth of the multiplicative Gaussian kernel

$$G(\Sigma_i, X(j) - S_1(i)) = \frac{1}{\sqrt{|\Sigma_i 2\pi^k}} \exp\left[-\frac{1}{2}(X(j) - S_1(i))\,\Sigma_i^{-1}\,(X(j) - S_1(i))^T\right]$$

The bandwidth for the individual kernels is calculated from S_1 using an expectation maximisation algorithm described in [68].

The distances between the S_1 and the distributions R and T are carried out in a Monte Carlo fashion (at each iteration another M samples are selected at random, using replacement, from S_2) until the null hypothesis can or cannot be rejected at significance p. Details of the Monte Carlo approach are given in [68].

To make use of the ChDF in a PLIS, we allow each record to serve as an observation and each sample point, or alternatively frequency bin, as a variable; S would represent the set of signals from a known device and T signals from a test device. It may prove necessary to use the frequency domain exclusively if the signals cannot be adequately aligned (while alignment is always possible, the voltage at a particular sample point may vary due to triggering errors). In any case, as the number of sample points is large (being proportional to the sample rate) and the ChDF algorithm has runtime $O\left(|S_1|k^2\right)$, it will be necessary to either perform dimension reduction (using statistical techniques or by trading-off a higher sampling rate for higher resolution) or by randomly selecting a subset of variables from S_1 to use with the distance metric each time a new T is tested. It must also be decided how the size of T affects proper identification; i.e., is it necessary to use one frame or many to determine the identity of a card?

CHAPTER 9. CONTRIBUTIONS

In this work we have: 1) delineated the major areas of study in physical layer identification and laid out a course of future work for researchers in these areas; 2) examined the body of literature that comprises the field of physical layer identification; 3) proposed a theoretically secure methodology for device identification and compared it with existing methodologies; 4) shown how the matched filter can be used to differentiate devices of the same model operating under the same channel conditions; 5) identified those portions of the 10Mb Ethernet signal most useful for device identification under a matched filter PLIS; 6) provided evidence that further research into PLI forensics is justified; 7) defined two types of attacks on PLIS; 8) proposed a framework to measure the resistance of a PLIS to an attack using an arbitrary waveform generator; 9) provided a way to estimate how many devices a PLIS is theoretically capable of distinguishing between; and 10) shown how an arbitrary component can be empirically modelled to determine how much it contributes to device variation.

APPENDIX A. MATLAB data acquisition routine

```
% NOTE: we assume that the scope has been correctly configured to capture
% 10Mb data; furthermore, this configuration has been saved to 'Setup 1'
% on the scope.  If 'Setup 1' does not contain the appropriate settings
% file, it may retrieved from the compact flash card accompanying the
% scope—it has been saved as 'dilon10mb_2_5gs.set'.  Having loaded the
% settings file, the setup should then be resaved to 'Setup 1'.

%
% CONNECTION VAR
%
% set hw address for connection to scope (if hw address has changed—i.e.,
% the scope is plugged into a different USB port—then refer to
% 'matlab_visa_connection.txt' to determine port settings
vu = visa('ni', 'USB0::0x0699::0x0401::C010098::INSTR');
% set transfer buffer to record length:
%   difference of query(vu,'DATA:START?') and query(vu,'DATA:STOP?')
%   ASIDE: should try to make this a multiple of bus width...
vu.InputBufferSize=280000;

%
% SESSION VAR
%
% number of records to capture
n = 10%10010;
% length of record
l = 280000;
% sample rate
s = 2.50*10^9;
```

```
29 % save directory (remember trailing slash)
30 recDir = 'D:\_DILON\new_recs\';
31
32 % open connection to scope
33 fopen(vu);
34 % configure scope for 10Mb capture (recall 'Setup 1'); may need to recall
35 % manually...probably safer to load manually
36 % fprintf(vu,'*RCL 1');
37 % determine y-increment (voltage scale)
38 yinc = str2num(query(vu,'WFMP:YMULT?'));
39 % set data collection points
40 fprintf(vu,'DATA:START 340001'); %beginning of record, in buffer of 1*10^6
41 fprintf(vu,'DATA:STOP 620000'); %end of record, in buffer
42
43 rec = zeros(1,1);
44 ch1 = zeros(1,1);
45 ch2 = zeros(1,1);
46 i = 1; %current rec cnt
47 badRecs = 0;
48
49 % begin timer
50 tic;
51 % instruct scope to take single measurement
52 fprintf(vu,'ACQ:STATE ON');
53 while(i <= n)
54     t = round(toc); % get current running time
55     disp(['Record: ' num2str(i) '/' num2str(n) ...
56         ' (%' num2str(i/n*100) ...
57         ', num. bad: ' num2str(badRecs) '); ' ...
58         'Elapsed time: ' num2str(floor(t/3600)) ' hr. ' ...
59         num2str(mod(floor(t/60),60)) ' min. ' ...
60         num2str(mod(t,60)) ' sec.']);
61
62     % wait for scope to finish measurement (check scope state every 1/10
63     % of a second)
```

```
64      while str2num(query(vu,'ACQ:STATE?')) == 1
65          pause(.10);
66      end
67
68      % get channel one data
69      fprintf(vu,'DATA:SOURCE CH1');
70      fprintf(vu,'CURVE?');
71      ch1 = binblockread(vu,'int8')';
72      % get channel two data
73      fprintf(vu,'DATA:SOURCE CH2');
74      fprintf(vu,'CURVE?');
75      ch2 = binblockread(vu,'int8')';
76
77      % to save time, trigger new measurement on scope
78      fprintf(vu,'ACQ:STATE ON');
79
80      % ensure that rec is good (link pulses do not fall much below 0V, so we
81      % check that sampled signal is very negative)
82      if min(ch1(6000:8000)) < -80 %should be well into sync signal by here
83          % recover differential signal
84          rec = yinc*(ch1 - ch2);
85
86          % write record to file
87          is = num2str(i);
88          f = [recDir 'sample' ...
89              strrep(num2str(zeros(1,5-length(is))),' ','') is '.mat'];
90          save(f,'s','rec');
91
92          % increment good recs counter
93          i = i + 1;
94
95      else % we have a bad record, save it and modify counters
96          badRecs = badRecs + 1;
97
98          % write raw channel data to file
```

```
99          badRecStr = num2str(badRecs);
100         f = [recDir 'bsample' badRecStr '.mat'];
101         save(f,'ch1','ch2','s','yinc');
102     end
103
104     % zero channel data to ensure we don't write duplicate records
105     ch1 = zeros(1,l);
106     ch2 = zeros(1,l);
107 end
108
109 % close connection to scope
110 fclose(vu);
```

APPENDIX B. Code for type one attack

Resolution

```
function [n, s_avg2] = getResolution2(s_avg,refSig,th_p,th_n,V_fs)
% find minimum number of bits (n) necessary to represent s_avg and still
% have filter output within th_p/th_n for an AWG with given V_FS
% n: resolution of AWG
% s_avg2: valid representation of s_avg using fewest number of bits

n = 14; %won't consider AWG with resolutions above 14 bits

% find possible voltage values given n and V_fs and round s_avg towards it
s_avg2 = s_avg;
while dot(refSig,s_avg2) >= th_n && dot(refSig,s_avg2) <= th_p
    % calc yinc based upon n and V_fs
    yinc = V_fs/(2^n-1);
    % possible voltage values given n and V_fs
    v = yinc*(-2^n:1:2^n-1)';
    % find which new voltage level each sample point of s_avg is closest to
    [~,I] = min(abs(bsxfun(@minus,s_avg',v)));

    s_avg_old = s_avg2; %won't know if we should stop until after we've
                        %overwritten valid s_avg2
    s_avg2 = v(I);
    n = n - 1;
end

% restore falsely decremented bit and need additional bit for sign
n = n + 2;
```

```
27 s_avg2 = s_avg_old;
```

Settling time

```
1  function [ts, tr, td] = getTs(s_avg,s_n,fs)
2  % find settingling time (ts) necessary to create signal between s_+/-
3  % assuming PLIS sampler with given fs
4  % ts: settling time of AWG
5  % tr: rise time of AWG
6  % td: dead time of AWG
7
8  l = length(s_avg);
9  T = 1/fs;
10
11 % set td... really is immaterial as we care about sum of tr and td
12 td = 0.01*T;
13 % percentage of s_avg AWG must reach at each sample point
14 p_s_avg = abs(s_n)./abs(s_avg);
15 % only care about max percentage (should always be less than one... doesn't
16 % make sense for it to be greater than one--an artifact of sampling)
17 m_p_s_avg = max(p_s_avg(p_s_avg < 1));
18
19 % max rise time for s_n
20 tr = (T-td)/(m_p_s_avg);
21 % settling time
22 ts = td + tr;
```

SNR

```
1  function [snr,snr_s,N_s_avg] = getSnr(refSig,s_avg,th_p,th_n)
2  % find SNR necessary to create signal with filter output outside of th_+/-
3  % snr: SNR of AWG in dB
4
5  % since we're generating random noise, we run this many times to be sure of
```

```
% the calculated snr
n = 1000;
% snr per iteration; average to find snr
SNR = zeros(n,1);

for i = 1:n
    snr_i = 30; %max SNR in dB
    snr_dec = 0.01; %amount to decrease SNR by until signal outside of th_+/-

    N_s_avg = s_avg; %our noisy signal
    m0 = dot(refSig,N_s_avg);

    while m0 >= th_n && m0 <= th_p
        snr_i = snr_i - snr_dec;
        N_s_avg = awgn(s_avg, snr_i,'measured');
        m0 = dot(refSig,N_s_avg);
    end

    SNR(i) = snr_i;
end

snr = mean(SNR);
snr_s = std(SNR);
```

THD

```
function [THD] = getThd(s_avg,s_p,s_n)
% find max THD that keeps signal within bounds set by A_+/-
% THD: THD in dB for each freq bin

S_avg = fft(s_avg);
D_p = fft(s_p-s_avg); %fft of distortion
D_n = fft(s_avg-s_n);

```

```
bw = 121; %index of maximum frequency for THD calcs; i.e. bandwidth over which
    THD is calculated
THD = zeros(bw-1,2); %THD is calculated on bin-by-bin basis
n_har = 9; %num of harmonics to use when calculating THD; six or ten depending
    upon standard (those include fundamenta)
n_bins = size(S_avg,1); %num of bins

for i = 2:bw %S_avg(1) is DC, which is excluded in THD calcs
    h_ind = (i-1)*[2:n_har+1]+1; %indicies of our harmonics

    % ignore denominator as we're taking ratios below
    P = (2*abs(S_avg(i)).^2); %power of undistorted fundamental freq
    P_p = 2*sum(abs(D_p(h_ind)).^2); %power of distorted harmonic
    P_n = 2*sum(abs(D_n(h_ind)).^2);

    THD(i-1,1) = 10*log10(P_p/P);
    THD(i-1,2) = 10*log10(P_n/P);
end
```

THD+N

```
function [THDN] = getThdN(refSig,s_avg,s_p,th_p,th_n)

% define (max) distortion
d_p = s_p-s_avg;

% d = d_inc*d_p
d_inc = 0.1;

S_avg = fft(s_avg);
D_p = fft(d_p); %fft of distortion

bw = 121; %index of maximum frequency for THD calcs; i.e. bandwidth over which
    THD is calculated
THDN = zeros(bw-1,1/d_inc); %THD is calculated on bin-by-bin basis
```

```
14 n_har = 9; %num of harmonics to use when calculating THD; six or ten depending
       upon standard (those include fundamenta)
15 n_bins = size(S_avg,1); %num of bins
16
17
18 for k = d_inc:d_inc:1
19     k_ind = round(1/(d_inc/k));
20     d = k*d_p;
21     D = fft(d);
22
23     parfor i = 2:bw %S_avg(1) is DC, which is excluded in THD calcs
24         i2 = i-2; %fundamental freq on right side
25         h_ind = (i-1)*[2:n_har+1]+1; %indicies of our harmonics; left side
26         h_ind2 = h_ind-2; %indicies of our harmonics; right side
27         h = [h_ind h_ind2]; %harmonics, both sides
28
29         %signal we'll add distortion to
30         SD = S_avg;
31         % only add in distortion components for bin under consideration
32         SD(h) = SD(h) + D(h); %signal plus distortion
33         sd = ifft(SD); %need time-domain for noise calcs
34         % add noise to distorted signal until outside of th_+/- (bw of
35         % noise is fs/2)
36         [snr,snr_s,n_sd] = getSnr(refSig,sd,th_p,th_n);
37         % need just the noise for calcs (fft, actually)
38         N = fft(n_sd - sd);
39
40         % ignore denominator as we're taking ratios below
41         P = 2*abs(S_avg(i))^2; %power of undistorted fundamental freq
42         P_p = 2*sum(abs(D_p(h_ind)).^2); %power of distorted harmonics
43         P_N = sum(abs(N).^2); %noise over bw=fs/2
44
45         THDN(i-1,k_ind) = 10*log10((P_p+P_N)/P);
46     end
47 end
```

SFDR

```
function sfdr = getSfdr2(s_avg,refSig,th_p,th_n)
% find SFDR ratio necessary to produce filter output for s_avg outside of
% thp/thn; bins included based on power
% s_avg: average of aligned records; attacker's signal constructed from
% this on a per bin basis
% refSig: ref sig of genuine device
% th_p/th_n: thresholds for filter output
% sfdr: n-by-1 vector of sfdr necessary to put s_avg outside of thresholds;
% each row corresponds to number of components used to construct attacker's
% signal

% num of components to consider
n = floor(size(s_avg,1)/2);
% sfdr: ratio of noise-to-signal for each of the i-components used in creating
% attacker's signal; 0 denotes that number of components used wasn't
% sufficient to create attacker signal within thresholds
sfdr = zeros(n,1);

S_avg = fft(s_avg);
l = size(S_avg,1);

% obtain indices of bin power sorted from greatest to least
[B,IX] = sort(abs(S_avg(1:floor(l/2))),'descend'); IX=IX';

% include i bins in attacker signal construction
parfor i = 1:floor(l/2)
    % attacker's signal, freq. domain
    S_att = zeros(l,1);

    % add individual components to attacker's signal
    if isempty(find(IX(1:i) == 1,1)) %if DC component is absent
        S_att([IX(1:i) end-IX(1:i)+2]) = S_avg([IX(1:i) end-IX(1:i)+2]);
    else %dc component present
```

```
34          %remove dc component index so it doesn't mess up right hand side index
                calcs
35          IX2 = IX(IX(1:i) ~= 1); %dc always at index one
36          S_att([1 IX2(1:i-1) end-IX2(1:i-1)+2]) = ...
37              S_avg([1 IX2(1:i-1) end-IX2(1:i-1)+2]);
38      end
39
40      % attacker's time domain signal using i most powerful bins
41      s_att = ifft(S_att);
42
43      % prepare to add noise to s_att
44      s_noi = s_att; %s_att with noise added to it
45      r = -100; %noise-to-signal component ratio (sfdr); start at -100 dB and
            increase
46
47      % check to see if attacker's signal falls within thresholds, if so then
48      % generate noise for each component and add it to attacker's signal
49      while dot(refSig,s_noi) >= th_n && dot(refSig,s_noi) <= th_p
50          S_noi = S_att;
51
52          % add noise to S_att for jth component
53          for j = 1:i
54              % power of jth component of S_att
55              if IX(j) == 1 %dc component only uses one bin
56                  p_j = abs(S_att(IX(j)))^2/l^2;
57              else
58                  p_j = 2*abs(S_att(IX(j)))^2/l^2;
59              end
60              % gen noise of sufficient power so that we reach ratio of r=n/s
61              noi = genNoise(10^(r/10)*p_j,l);
62              % where we'll put the noise (placed randomly)
63              noi_ind = randi([1 floor(l/2)]);
64              % add noise to s_att
65              S_noi(noi_ind+1) = S_noi(noi_ind+1) + noi; %left side
66              S_noi(end-noi_ind+1) = S_noi(end-noi_ind+1) + noi'; %right side
```

```
67          end
68
69          s_noi = ifft(S_noi);
70          r = r + 1;
71      end
72
73      sfdr(i) = r-1;
74  end
75
76
77  function n = genNoise(p,l)
78  % generate complex noise of power p
79
80  x = sqrt(p*l^2/2)*rand(1); %real part of noise
81  y = sqrt(p*l^2/2-x^2); %complex part of noise
82
83  % determine sign of each component
84  if rand(1) < 0.5
85      x = -1*x;
86  end
87
88  if rand(1) < 0.5
89      y = -1*y;
90  end
91
92  n = x + 1j*y;
```

BIBLIOGRAPHY

[1] R. Jones, *Most Secret War*. Hamilton, 1978.

[2] D. L. Mensa *et al.*, "Radar signature evaluation apparatus," United States Patent 6,529,157, March 2003.

[3] T. L. Overman and K. C. Overman, "Adaptive radar threat detection and tracker verification system," United States Patent 4,146,892, March 1979.

[4] P. J. Ferrell, "Method and apparatus for characterizing a radio transmitter," United States Patent 5,005,210, April 1991.

[5] M. B. Frederick, "Cellular telephone anti-fraud system," United States Patent 5,448,760, September 1995.

[6] W. G. Barrere, D. Kaplan, and E. R. Green, "Adaptive waveform matching for use in transmitter identification," United States Patent 5,715,518, February 1998.

[7] D. N. Hoogerwerf, E. R. Green, D. M. Stanhope, and R. Mickernan, "Active waveform collection for use in transmitter identification," United States Patent 6,035,188, March 2000.

[8] D. Kaplan and D. M. Stanhope, "Waveform collection for use in wireless telephone identification," United States Patent 5,999,806, December 1999.

[9] M. J. Marcus, "Progress in vhf/nhf mobile transmitter identification," University of Manitoba, Department of Electrical and Computer Engineering, Tech. Rep., 1992, technical Report.

[10] J. Toonstra and W. Kinsner, "Transient analysis and genetic algorithms for classification," in *Proceedings of the IEEE Conference on Communications, Power, and Computing (WESCANEX 95)*, vol. 2, 1995, pp. 432–437.

[11] Y. Payal, "Identification of push-to-talk transmitters using wavelets," Master's thesis, Naval Postgraduate School, Monterey, CA, 1995.

[12] R. D. Hippenstiel and Y. Payal, "Wavelet based transmitter identification," in *Proceedings of the Fourth International Symposium on Signal Processing and Its Applications (ISSPA 96)*. IEEE Computer Society, 1996.

[13] J. Toonstra and W. Kinsner, "A radio transmitter fingerprinting system odo-1," in *Proceedings of the 1996 Canadian Conference on Electrical and Computer Engineering*. IEEE Computer Society, 1996, pp. 60–63.

[14] D. Shaw and W. Kinsner, "Multifractal modeling of radio transmitter transients for classification," in *Proceedings of the IEEE Conference on Communications, Power, and Computing (WESCANEX 97)*. IEEE Computer Society, 1997, pp. 306–312.

[15] K. J. Ellis and N. Serinken, "Characteristics of radio transmitter fingerprints," *Radio Science*, vol. 36, pp. 585–598, 2001.

[16] J. Kamarainen, V. Kyrki, and T. Lindh, "Signal discrimination based on power spectrum of filter response," Lappeenranta University of Technology, Department of Information Technology, Tech. Rep. 80, 2002, research Report.

[17] O. Ureten and N. Serinken, "Detection of radio transmitter turn-on transients," *Electronics Letters*, vol. 35, no. 23, pp. 1996–1997, November 1999.

[18] J. Hall, M. Barbeau, and E. Kranakis, "Detection of transient in radio frequency fingerprinting using signal phase," in *Proceedings of the 3rd IASTED International Conference on Wireless and Optical Communications (WOC 2003)*. ACTA Press, 2003, pp. 13–18.

[19] ——, "Enhancing intrusion detection in wireless networks using radio frequency fingerprinting," in *Proceedings of Communications, Internet and Information Technology (CIIT 2004)*. ACTA Press, 2004.

[20] E. A. Jackson, "Detecting intrusions at layer one: device fingerprinting for network access authorization," Master's thesis, Iowa State University, Ames, IA, 2006.

[21] M. Mina, T. E. Daniels, S. F. Russell, and R. M. Gerdes, "Intrusion detection, performance, assurance, and system maintenance: A new paradigm in computer security," *Materials Evaluation*, vol. 63, no. 12, pp. 1203–1218, 2005.

[22] T. E. Daniels, M. Mina, and S. F. Russell, "A signal fingerprinting paradigm for physical layer security in conventional and sensor networks," in *Proceedings of the First IEEE/Createnet International Conference on Security and Privacy for Emerging Areas in Communnication Networks (SecureComm)*. IEEE Computer Society, 2005, pp. 219–221.

[23] J. Hall, M. Barbeau, and E. Kranakis, "Detecting rogue devices in bluetooth networks using radio frequency fingerprinting," in *Proceedings of the Third IASTED International Conference on Communications and Computer Networks*. IASTED/ACTA Press, 2006, pp. 108–113.

[24] N. Saparkhojayev and D. R. Thompson, "Matching electronic fingerprints of rfid tags using the hotelling's algorithm," in *Proceedings of the IEEE Sensors Applications Symposium (SAS 2009)*. IEEE Computer Society, 2009, pp. 19–24.

[25] O. Ureten and N. Serinken, "Wireless security through rf fingerprinting," *Canadian Journal of Electrical and Computer Engineering*, vol. 32, pp. 27–33, 2007.

[26] K. B. Rasmussen and S. Capkun, "Implications of radio fingerprinting on the security of sensor networks," in *Proceedings of the Third International Conference on Security and Privacy in Communications Networks and the Workshops (SecureComm 2007)*. IEEE Computer Society, 2007, pp. 331–340.

[27] I. Kennedy, P. Scanlon, F. Mullany, M. Buddhikot, K. Nolan, and T. Rondeau, "Radio transmitter fingerprinting: A steady state frequency domain approach," in *Proceedings of the IEEE 68th Vehicular Technology Conference (VTC 2008)*. IEEE Computer Society, 2008, pp. 1–5.

[28] M. Hamdi, A. Meddeb-Makhlouf, and N. Boudriga, "Multilayer statistical intrusion detection in wireless networks," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, 2009.

[29] L. Xiao, L. Greenstein, N. Mandayam, and W. Trappe, "Using the physical layer for wireless authentication in time-variant channels," *IEEE Transactions on Wireless Communications*, vol. 7, pp. 2571–2579, 2008.

[30] B. Danev, T. S. Heydt-Benjamin, and S. Capkun, "Physical-layer identification of rfid devices," in *Proceedings of the USENIX Security Symposium (USENIX-SS 09)*. USENIX Association, 2009, pp. 199–214.

[31] J. W. Erbskorn, "Detection of intrusions at layer one: The ieee 802.3 normal link pulse as a means of host-to-network authentication a preliminary performance analysis and survey of environmental effects," Master's thesis, Iowa State University, Ames, IA, 2009.

[32] B. Danev and S. Capkun, "Transient-based identification of wireless sensor nodes," in *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks (IPSN 09)*. IEEE Computer Society, 2009, pp. 25–36.

[33] H. Mustafa, M. Doroslovacki, and H. Deng, "Automatic radio station detection by clustering power spectrum components," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 02)*, vol. 4. IEEE Computer Society, 2002, p. 4168.

[34] S. Xu, L. Xu, Z. Xu, and B. Huang, "Individual radio transmitter identification based on spurious modulation characteristics of signal envelop," in *Proceedings of the Military Communications Conference (MILCOM 2008)*. IEEE Computer Society, 2008, pp. 1–5.

[35] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proceedings of the 14th ACM international conference on Mobile computing and networking (MobiCom 08)*. ACM, 2008, pp. 116–127.

[36] A. Candore, O. Kocabas, and F. Koushanfar, "Robust stable radiometric fingerprinting for wireless devices," in *Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust*, vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, 2009, pp. 43–49.

[37] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," in *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2005, pp. 211–225.

[38] S. J. Murdoch, "Hot or not: revealing hidden services by their clock skew," in *Proceedings of the 13th ACM conference on Computer and communications security (CCS '06)*. ACM Press, 2006, pp. 27–36.

[39] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. V. Randwyk, and D. Sicker, "Passive data link layer 802.11 wireless device driver fingerprinting," in *Proceedings of the 15th conference on USENIX Security Symposium (USENIX-SS 06)*. USENIX Association, 2006.

[40] D. L. C. Choong, C. C. Yuan, T. C. Pheng, and R. S. Lee, "Identifying unique devices through wireless fingerprinting," in *Proceedings of the first ACM conference on Wireless network security (WiSec 08)*. ACM Press, 2008, pp. 46–55.

[41] B. Sieka, "Active fingerprinting of 802.11 devices by timing analysis," in *Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC 2006)*. IEEE Computer Society, 2006, pp. 15–19.

[42] S. Bratus, C. Cornelius, D. Kotz, and D. Peebles, "Active behavioral fingerprinting of wireless devices," in *Proceedings of the first ACM conference on Wireless network security (WiSec 08)*. ACM Press, 2008, pp. 56–61.

[43] P. S. Ravikanth, "Physical one-way functions," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[44] B. Gassend, D. Clarke, M. Dijk, and S. Devadas, "Delay-based circuit authentication and applications," in *Proceedings of the 2003 ACM symposium on Applied computing*, ser. Computer security. ACM Press, 2003, pp. 294–301.

[45] B. Gassend, D. E. Clarke, M. Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the ACM Conference on Computer and Communications Security*. ACM Press, 2002, pp. 148–160.

[46] ——, "Controlled physical random functions," in *Proceedings of the Annual Computer Security Applications Conference (ACSAC 02)*. IEEE Computer Society, 2002, p. 149.

[47] B. Danev, H. Luecken, S. Capkun, and K. E. Defrawy, "Attacks on physical-layer identification," in *Proceedings of the third ACM conference on Wireless network security (WiSec '10)*, vol. 0. New York, NY, USA: ACM, 2010, pp. 89–98.

[48] M. Edman and B. Yener, "Active attacks against modulation-based radiometric identification," Rensselaer Polytechnic Institute, Department of Computer Science, Tech. Rep., 2009, technical Report.

[49] R. M. Bolle, J. H. Connell, S. Pankanti, N. K. Ratha, and A. W. Senior, *Guide to Biometrics*. Springer, 2004.

[50] R. M. Gerdes, T. E. Daniels, M. Mina, and S. F. Russell, "Device identification via analog signal fingerprinting: A matched filter approach," in *Proceedings of the 2006 Network and Distributed System Security Symposium (NDSS 2006)*. The Internet Society, 2006.

[51] G. J. Hahn and W. Q. Meeker, *Statistical Intervals: A Guide for Practitioners*. Wiley, 1991.

[52] L. Couch, *Digital and Analog Communication Systems*. Macmillan Publishing Company, 1993.

[53] C. L. Phillips, J. M. Parr, and E. A. Riskin, *Signals, Systems and Transforms*. Prentice Hall, 2003.

[54] C. Chatfield, *Principles of Forecasting: A Handbook for Researchers and Practitioners*, ser. International Series in Operations Research & Management Science. Springer, 2001, vol. 30, ch. Prediction Intervals for Time-Series Forecasting, pp. 475–494.

[55] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-IEEE, 2004.

[56] R. Kohavi and F. Provost, "Glosssary of terms," *Machine Learning*, vol. 30, pp. 271–274, 1998.

[57] C.-E. Särndal, B. Swensson, and J. Wretman, *Model Assisted Survey Sampling*, ser. Springer Series in Statistics. Springer, 2003.

[58] IEEE, "Ieee 802.3-2008 ieee standard for information technology-specific requirements– part 3: Carrier sense multiple access with collision detection (cmsa/cd) access method and physical layer specifications," IEEE, Tech. Rep., 2008, iEEE Std 802.3–2008.

[59] M. Burns and G. W. Roberts, *An introduction to mixed-signal IC test and measurement*. Oxford University Press, 2001.

[60] W. Kester, "Evaluating high speed dac performance," Analog Devices, Tech. Rep., 2008, mT-013 Tutorial.

[61] W. Kester, Ed., *The data conversion handbook*. Newnes, 2005.

[62] E. Balestrieri, "Some critical notes on dac time domain specifications," in *Proceedings of the IEEE Instrumentation and Measurement Technology Conference (IMTC 2006)*. IEEE Computer Society, 2006, pp. 930–935.

[63] H. K. Schoenwetter, "High-accuracy settling time measurements," *IEEE Transactions on Instrumentation and Measurement*, vol. 32, no. 1, pp. 22–27, March 1983.

[64] IEEE, "Ieee standard for terminology and test methods for analog-to-digital converters," IEEE, Tech. Rep., 2011, iEEE Std 1241–2010.

[65] R. J. Weber, *Introduction to Microwave Circuits: Radio Frequency and Design Applications*. IEEE Press, 2001.

[66] S. Kumar and K. K. S. Suresh, *Electric Circuits and Networks*. Pearson Education, 2009.

[67] I. Valor, "Fl1137 rev k," Valor, Inc., Tech. Rep., N/A, datasheet.

[68] X. Song, M. Wu, C. Jermaine, and S. Ranka, "Statistical change detection for multi-dimensional data," in *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2007, pp. 667–676.