A Network-based Approach to Counterfeit Detection

Supreeth Sathyanarayana
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332
Email: supreeth5@gatech.edu

William H. Robinson
Department of Electrical Engineering and Computer Science
Vanderbilt University
Nashville, TN 37235
Email: william.h.robinson@vanderbilt.edu

Raheem A. Beyah
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332
Email: rbeyah@ece.gatech.edu

***Abstract*—Counterfeit devices are wreaking havoc in the industry today and are causing billions of dollars of loss in revenue to companies. These devices usually have fake components or have components that are re-marked as better ones. Illegitimate workshops even manufacture counterfeit devices on a large scale. Hence, counterfeit detection is of utmost importance. However, most counterfeit detection techniques today have complex and expensive setups that are slow in generating results. Some methods are also destructive to the device under test, and can hence only be applied to a small portion of the suspect devices. Most methods are also usually intended to target a specific set of devices. In this paper, we propose a simple and cheap technique of counterfeit detection, which we believe is a first-of-its-kind, network-based solution. Being network-based, it can be used to swiftly test a broad range of networked devices. The technique only monitors the network traffic of the device, therefore it is non-destructive. We first illustrate the general efficacy of the technique using FPGAs. Next, we show that lower-end processors (i.e., Core i3) in real systems can be differentiated from higher-end processors (i.e., Core i7) based on the host node's network traffic. This technique is effective against devices with counterfeit components or even with legitimate, but re-marked (as higher capacity) components. Using a neural network based classifier, we show classifier recall values (i.e., the ratio of the number of true positives to the sum of the number of true positives and false negatives) of up to 78.7% using traffic captures of 2,500 packets.**

I. Introduction

Counterfeiting is the forging or imitating of a legitimate device by an illegitimate device, usually of lower capacity or quality, so as to afford the seller a higher profit margin through a lower manufacturing cost. Counterfeiting of devices is widespread in the industry today [1], [2], and counterfeit devices are a common bane to companies and governments alike [3], [4]. Counterfeit devices amount to billions of dollars of loss in revenue to US semiconductor companies [5], [6]. Counterfeiting also impacts the customer, since counterfeit devices usually have either old, repackaged versions of the components, or components made from lower quality materials. Hence, the customer receives a degraded, outdated version of the device, or a low-quality, less reliable version of the device. This impacts no customer more than the military, whose critical systems have recently been found to be compromised by counterfeit components [1], [7], [8], [9]. The resulting impact on national security can be devastating, not to mention the risk to the lives of the operators of lethal equipment with counterfeit, unreliable components.

Companies and governments are in turn, spending huge amounts of time and money to detect, deter, and counter these counterfeit devices [4], [8], [10], [11]. Thus, counterfeit devices are widespread, and counterfeit detection is of very high importance. There are many counterfeit detection techniques available today [5], [12], [13]. However, most of the techniques available are either expensive, complex, slow, destructive to the device under test (DUT), narrow-scoped, or require specialized hardware. There is thus, a definite need for a more efficient and effective counterfeit detection mechanism, without the drawbacks of the existing methods.

In this paper, we propose a technique that is simple, cheap, fast, non-destructive, and broad-scoped, which is also a first-of-its-kind network-based solution. Our technique is based on using the interarrival times (IATs) of network traffic to fingerprint devices, similar to works like [14], [15], [16], [17], [18], [19], but we go a step further and fingerprint the device components themselves (both legitimate and illegitimate), thereby enabling us to detect the replacement of a legitimate device component with an illegitimate one. We first perform experiments with field-programmable gate arrays (FPGAs) to show the effect that device components have on the network traffic of the device. We emulate network-enabled computer systems on the FPGA board. Then, we capture and statistically analyze the network traffic IATs of different traffic types (i.e., UDP, TCP, ICMP). We then change just the processor configuration of the emulated system and repeat the traffic captures and analyses. We show how the IAT statistics are consistently different from the original configuration. This is repeated on different FPGA boards to illustrate the generality of the technique.

Armed with this understanding of the effect of device components on the IATs of network traffic, we repeat similar experiments on a real computer with an Intel Core i7 processor. We then change the processor to an Intel Core i3 and repeat the captures. The IATs of the i7- and i3-based systems are used to train a neural network and classify the i3 as a counterfeit i7. Hence, this technique can be used to detect a counterfeit device component that is illegally re-marked as a legitimate one [20]. The recent discovery of a counterfeiting workshop [21] manufacturing counterfeit devices on a large scale only establishes the relevance and global scope of this threat. Our technique is geared towards this type of counterfeiting, where the device has counterfeit components or has re-marked components [22] of lower capacity. Once we obtain the signatures of these counterfeit components, our technique can be used to differentiate counterfeit signatures from legitimate signatures.

The rest of this paper is organized as follows. Section II discusses the related work and how our contribution compares. Section III describes the experimental setup and our technique in greater detail. Section IV discusses the FPGA-based experiments that form the basis for our technique. In Section V, we show the actual counterfeit detection results from the real computer experiments. Section VI discusses the results and some limitations of our technique, and finally, Section VII provides the conclusion and future work.

II. Related Work

There are as many counterfeit detection techniques today as there are counterfeiting methods themselves [23]. In intellectual property (IP) watermarking [24], a hidden signature is embedded in the design. In the field, this signature is then used to confirm the legitimacy of the design. However, this method requires intervention in the manufacturing process to insert the watermarks. Our technique does not alter the manufacturing process, and instead relies on the physical uniqueness of each device component to identify the legitimate device.

Hardware metering and auditing involves observing some unique characteristic of the integrated circuits (ICs) of the device. There are two types of hardware metering, passive [25], [26], [27] and active [28]. Passive hardware metering involves observing an identifying quality of the IC without modifying it in anyway, while active hardware metering requires the addition of extra logic to the IC to make it identifiable. While the drawback of the active variant is that the manufacturing process needs to be modified, the drawback of the passive variant is that it is expensive, as all the logic of the IC needs to be characterized with high precision. This becomes even more prohibitive when scaled to large ICs, as the linear equations to be solved become impractical. The technique we propose is like the passive hardware metering in that it requires no modification to the device, however, it does not require any expensive or complex characterization and processing.

The physical unclonable function (PUF) [29], [30], [31] is a physical function that provides a unique mapping from its inputs to its outputs based on the unique variations of the unclonable characteristics of the device material, such as current and timing. This technique modifies the manufacturing design to include the PUFs. Hence, these methods either need intervention during the manufacturing process, or need specialized/expensive equipment for counterfeit detection. Being network-based, our technique requires no modification to the DUT. It only needs a simple computer to capture the device's network traffic and determine its components' legitimacy.

Benchmarking software like [32] can provide a detailed picture of the system and any changes to its components. Any lower capacity component substituted for the legitimate one in the device will lead to sub-optimal performance on the benchmark tests and hence flag the presence of counterfeit components. However, such an approach requires that the benchmarking software be available for, then installed and run on every suspect device, which is not scaleable. The technique we propose makes no changes to the system beyond the sending and receiving of network packets.

Fingerprinting a device is another approach that has been explored for counterfeit detection. This involves forming a signature that is based on some unique physical characteristic of the device, which is then compared with the DUT to verify its authenticity. The work in [12] deals with a fingerprinting technique based on unique radio frequency (RF) emissions from the device, which are recorded and matched against a fingerprint database. This is afforded by the fact that every device emits unique electromagnetic radiation. However, collecting and processing this radiation-based fingerprint requires expensive and complex hardware setup. Similarly, the technique in [33] uses X-Rays to analyze the authenticity of the device based on its physical structure at a microscopic level using an expensive X-Ray inspection system. Our technique fingerprints the device based on its packet interarrival delays. Hence, we only need a computer connected to the network to extract the signature of the device.

III. Background, Experimental Setup, and Technique

This section describes the inspiration for our technique, how we implement it for counterfeit detection, and the general setup used for both the FPGA and real computer experiments.

A. Background

The process to create network packets is complex and influenced by several factors. Each packet is formed of multiple layers wrapped one within another [34]. Many components of the computer (e.g., processor, cache, main memory) are involved in the packet's formation, and it is intuitive to think that each of them will have an influence on how the packet is formed. The author of [34] has in fact shown how a *slow* receiver affects the traffic dynamics differently than a *fast* receiver. This hints at a certain dependence of internal hardware architecture on the packet delays. This can be exploited to create unique IAT-based fingerprints for a legitimate device configuration, which can then be contrasted with fingerprints of devices modified with counterfeit components.

B. Experimental Setup and Technique Overview

A monitor node captures the traffic from the DUT via a network tap (nTAP) as shown in Figure 1. The sender/receiver (Sony VAIO laptop) is used to generate traffic to the DUT for active (ICMP echo requests to the device) tests and to receive the DUT's traffic for passive (TCP, UDP, and ICMP traffic from the device) tests.

The monitor node runs all the necessary scripts to control the VAIO and the DUT. It controls the sending of ICMP requests from the VAIO to the DUT, and the sending of TCP, UDP, and ICMP packets from the DUT to the VAIO. All packets leaving the DUT are captured via the nTAP by a NetFPGA 1G [35] installed in the monitor node. Regular network interface controllers (NICs) interrupt the system for each packet that arrives, and these interrupts are serviced either instantly or in groups by the operating system (OS), which then timestamps the packets using the system clock. This leads to less precise timestamps. Hence, we use a NetFPGA (a reconfigurable hardware platform for high-speed networking loaded with a packet generator program) to provide accurate hardware timestamping as low as 8 ns with its 125 MHz core clock. Once the packets are captured and timestamped, the monitor node filters the packets using tshark, extracts the IATs using Matshark [36] in MATLAB, and generates probability distribution functions (PDFs) of the IATs.

Before using this setup to generate PDFs of IATs and perform counterfeit detection on a real computer, we first test the repeatability and stability of the IAT-based signatures of devices emulated on FPGAs. This is done in Section IV, wherein the FPGAs provide a realistic representation of a device while also affording a high degree of control over

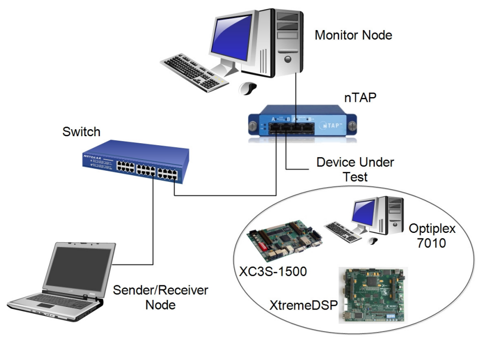

Fig. 1: Testbed to capture and process traffic to and from DUT.

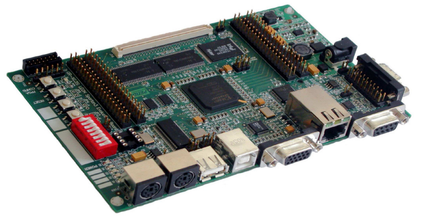

Fig. 2: Pender GR-XC3S-1500 Development Board.

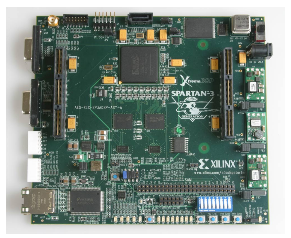

Fig. 3: Xilinx XtremeDSP Starter Platform - Spartan-3A DSP 1800A Edition.

the device component configuration. We vary just one component of the emulated device at a time and collect traffic IATs to bring forth the statistical separation in the PDFs of the different configurations. We also generate PDFs multiple times for each emulated device configuration to verify the signature's stability. Once the IAT-based signature is proven stable and differentiable, we perform similar experiments on a real computer in Section V. Here, we change the computer's processor and then use the inherent statistical separation in the PDFs of the collected traffic IATs to classify the processor as either legitimate or illegitimate.

IV. FPGA Experiments and Results

We aimed to measure network traffic from devices whose configurations only differed from each other by the processor used, and hence bring out the effect that the processor has on network traffic. Thus, we chose to emulate these devices on FPGAs, which provide high reconfigurability while also providing a faithful representation of the idiosyncrasies of real device components. These emulated devices were then loaded with a variant of Linux and used to send and receive different types of traffic, both active (ICMP requests) and passive (TCP, UDP). The advantage of the active technique compared to the passive one is that no interaction (i.e., a script that resides on the DUT to have it send traffic) with the DUT is required.

A. Emulated Device Under Test

Here we discuss the different hardware and software used to emulate the different DUT configurations on the FPGAs.

1) Pender GR-XC3S-1500 Development Board: The GR-XC3S-1500 Development Board [37] (Figure 2) features a Xilinx Spartan3 XC3S1500-FG456 FPGA, 64 MB SDRAM of on-board memory, Ethernet 10/100 Mbit/s MAC and PHY (LXT971A), a JTAG programming and configuration port, and 25 MHz and 50 MHz on-board oscillators.

2) Xilinx XtremeDSP Starter Platform - Spartan-3A DSP 1800A Edition: The XtremeDSP Starter Platform [38] (Figure 3) features a Xilinx 3SD1800A-FG676 FPGA, 128 MB DDR2 SDRAM of on-board memory, Ethernet 10/100/1000 Mbit PHY, a JTAG programming and configuration port, and a 125 MHz LVTTL SMT on-board oscillator.

3) LEON3 Processor: The LEON3 [39] is a synthesisable VHDL model of a SPARC-V8 based 32-bit processor. It has a fully pipelined IEEE-754 floating-point unit (FPU); hardware multiply, divide, and multiply-accumulate (MAC) units; and highly configurable, separate data and instruction cache (Harvard Architecture). Its VHDL model source code is distributed as part of the GRLIB IP library and is freely available under the GNU GPL license. The GRLIB IP is configured and downloaded onto the FPGAs using Xilinx ISE for each device configuration using a JTAG cable.

4) SnapGear: A special LEON version of the SnapGear Embedded Linux distribution, provided by Aeroflex Gaisler, is installed on the LEON3-based system that we synthesized on the FPGAs. This provides a simple yet potent version of Linux for use on the emulated systems. It includes all the basic functionalities of a Linux system, such as Ethernet networking support, and BusyBox [40] utilities. It is configured and downloaded onto the LEON3-based system in the FPGAs using a JTAG cable and the GRMON debugger [41].

B. Experimental Results

For each FPGA configuration, we generated four 15-minute captures of ICMP replies (Ping), TCP (Iperf), and UDP (Iperf) packets received from the DUT. The capture duration provided 2,500 packets. The FPGA configurations differed by just one component at a time. The components varied were: (1) processor clock speed, (2) processor data cache replacement policy, and (3) processor instruction cache replacement policy. The entire experiment was done for both of the FPGA boards. The PDFs were then plotted with 1,000 bins of 10 μs width. The default design that we emulated was a system composed of a LEON3 processor with Memory Management Unit (MMU), 40 MHz clock, 64 MB RAM for the GR-XC3S-1500 board and 128 MB RAM for the XtremeDSP board, 8 KB data cache, 4 KB instruction cache, and running SnapGear Linux. We performed our experiments with both boards and observed similar results. Due to space constraints, we only show results for the GR-XC3S-1500 board.

1) Processor Clock Speed: After capturing traffic for the default configuration, we changed the configuration by varying the clock speeds.

UDP packets of fifty six bytes each were generated on the DUT at 1 Mbps using Iperf. Figure 4 shows the PDFs of the IATs for the four captures where the DUT's clock speed was 40 MHz. As can be observed by the figure, the IAT PDFs are nearly identical. This similarity exists for all the configurations, thus subsequent figures will have a single line per configuration for the sake of presentation clarity. Figure 5 shows the variations in the PDFs of the IATs for clock speeds of 40 MHz, 33 MHz, 25 MHz, and 20 MHz. There is a lateral shift between the peaks of the different clock speeds, showing that increasing clock speed decreases the mean IATs. There is also a decrease in the heights of the PDF peaks with decreasing clock speeds, which leads to more spread of the IAT values along the x-axis for the lower clock speeds.

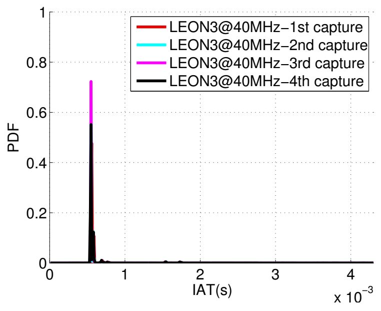

Fig. 4: PDFs of UDP IATs for four captures of 40 MHz clock on the GR-XC3S-1500 board.

Additionally, TCP data was generated using Iperf on the DUT. Figure 6 shows the PDFs of the IATs for clock speeds of 40 MHz, 33 MHz, 25 MHz, and 20 MHz. Here again, we see a shift to the right with decreasing clock speeds. This indicates that the configurations at the lower clock speeds were unable to match the peak TCP rate and hence had higher mean IATs. The decreasing heights of the peaks with decrease in clock speeds also indicates that the configurations at the lower clock speeds struggled to maintain a constant high data rate, thereby producing a more spread out PDF, with lower peaks. The dual peaks for each of the clock speeds also indicates two distinct rates at which the DUT was able to sustain the TCP traffic.

To evaluate this technique under active probing, the DUT was sent ICMP requests of 56 bytes per packet from the VAIO every 1 ms, and the replies were captured by the monitor node. Figure 7 shows the PDFs of the IATs for clock speeds of 40 MHz, 33 MHz, 25 MHz, and 20 MHz. The higher clock speeds, being able to match the required 1 ms IAT (i.e., the rate at which ICMP requests were generated) with more ease, have higher peaks and less spread of PDFs, while the lower clock speed PDFs are more spread out.

The DUT was also sent ICMP requests of 1400 bytes per packet from the VAIO every 1 ms. Figure 8 shows the PDFs of the IATs for clock speeds of 40 MHz, 33 MHz, 25 MHz, and 20 MHz. Here too, the higher clock speeds PDF peak is closer to the required 1 ms mean IAT (echo request send rate was 1 per 1 ms), while that of lower clock speeds is farther. When a device is interrupted by an ICMP request, the processor has to switch state and service the request. Hence, the slower clock speed configurations, which take longer to switch state and service the ICMP requests, have higher mean IATs.

2) Processor Data Cache Replacement Policy: To see the effects of data cache on the IATs, we emulated a configuration with 40 MHz clock and 8 KB data cache, and changed the data cache replacement policy from least recently used (LRU) to random replacement (RR), and captured different traffic types.

Figure 9 shows the PDFs for IATs of ICMP (1400 bytes) replies. The ICMP requests were sent every 1 ms. Here, the LRU policy configuration has lower mean IATs. However, its PDFs are more spread out while that of RR are more peaked, with a higher mean IAT.

3) Processor Instruction Cache Replacement Policy: To see the effects of instruction cache on network traffic, we emulated a configuration with 40 MHz clock and 8 KB instruction cache, and changed the instruction cache replacement policy from LRU to RR, and captured different traffic types.

Traffic that relied on the instruction cache, such as TCP and UDP, showed a marked shift in the RR PDFs to higher mean IATs. This is evident from Figure 10, which shows the PDFs for IATs of UDP (56 bytes per packet at 1 Mbps) packets. This is due to the instructions that are executed by the processor to run Iperf to generate UDP and TCP packets, which depends on the instruction cache replacement policy.

C. Discussion

We find that all traffic types are affected by changes to the processor clock. This is expected since the processor clock is a crucial part of the processor that decides how fast a packet or any instruction for that matter, is processed. Thus, we see IATs with higher mean values with decreasing clock speeds.

The data cache of the processor TCP, UDP, and ICMP (1400 bytes) traffic (because they make data accesses). Hence, any change in the data cache replacement policy caused a change in the performance of the device that was noticeable in its network traffic. ICMP (1400 bytes) had the most deviation in PDFs due to its high data payload. ICMP (56 bytes) packets were the least affected since they make very little use of the data cache; hence, changes to the data cache replacement policy had very little effect on the IATs.

The instruction cache of the processor affected TCP and UDP more than it did ICMP. This is because TCP and UDP packets were generated on the DUT using Iperf, which required the processor to step through many instructions. The rate at which these instructions were processed was affected by the instruction cache replacement policy. When comparing the PDF peaks of the RR configurations to the LRU configurations, we found a right shift to higher mean values of the IATs. For ICMP traffic, the mean values of the IATs were the same for both replacement policies, but the RR policy had a PDF with a greater spread and shorter height.

Thus, we can conclude that the effects of processor on the device's network traffic IATs are very consistent and noticeable, since it is a very crucial part of a computer system and will cause instructions (hence, packets sent or received) to be executed with different but consistent delay distributions. Variations made to the processor (e.g., clock speed, data cache replacement policy, or instruction cache replacement policy) are instantly differentiable from the original configuration by analyzing UDP, TCP, and ICMP traffic IATs. This lends itself very well to counterfeit detection, since crucial components like the processor are usually the most expensive in a system and hence most susceptible to counterfeiting.

V. Real System Experiments and Results

We used the same testbed and setup (Figure 1) as was used for the FPGAs in the previous section. The DUT was a real computer. The computer used was a Dell Optiplex 7010 with an Intel Core i7-3770 3.4 GHz processor, 6 GB RAM, running Ubuntu 11.10 version of Linux. The computer's ICMP reply traffic (56 bytes and 1400 bytes, every 1 ms), ICMP request traffic (56 bytes and 1400 bytes, every 1 ms), UDP traffic (56 data bytes at 1 Mbps), and TCP traffic (rate not controlled) were captured, and the IATs were analyzed statistically as before. All captures were four 15-minute captures for each traffic type except for TCP, whose high data rate allowed us to use four 5-minute captures to obtain approximately the same number of blocks of 2,500 packets for analysis as the other traffic types. These tests were repeated with different processors, and their traffic IATs were processed by a neural network based classifier (discussed in the next section).

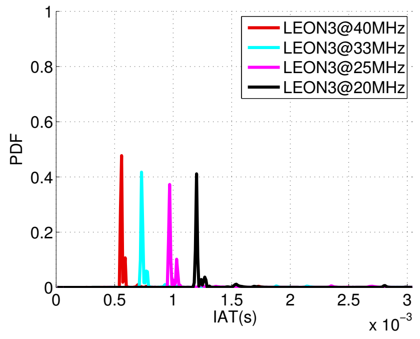

Fig. 5: PDFs of UDP IATs for clock variations on the GR-XC3S-1500 board.

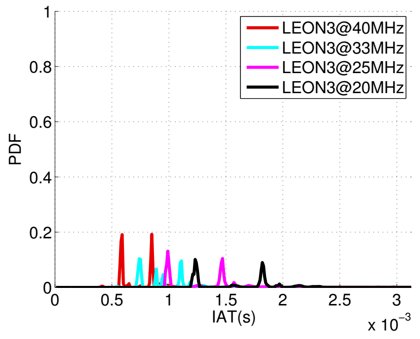

Fig. 6: PDFs of TCP IATs for clock variations on the GR-XC3S-1500 board.

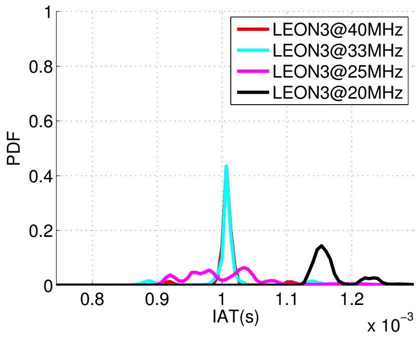

Fig. 7: PDFs of ICMP (56 bytes) reply IATs for clock variations on the GR-XC3S-1500 board.

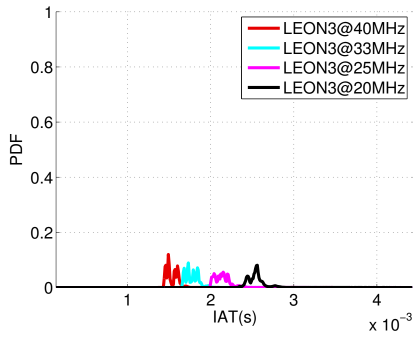

Fig. 8: PDFs of ICMP (1400 bytes) reply IATs for clock variations on the GR-XC3S-1500 board.

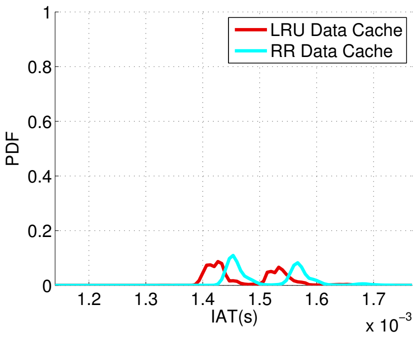

Fig. 9: PDFs of ICMP (1400 bytes) reply IATs for processor data cache variations on the GR-XC3S-1500 board.

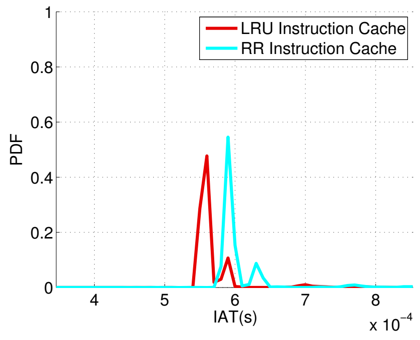

Fig. 10: PDFs of UDP IATs for processor instruction cache variations on the GR-XC3S-1500 board.

A. *Neural Network Based Classifier*

Neural networks, or specifically, artificial neural networks (ANNs), are mathematical models derived from and inspired by biological networks of neurons. They are used to model relationships between inputs and outputs. The relationships are modeled as a series of interconnections between neurons, which accept an input, operate on it based on some function, and then produce an output. Thus, the entire ANN can be thought of as a compound of all the functions represented by the individual neurons, taking in an input and producing an output based on the input. Just like the biological nervous system that they mimic, these ANNs can be trained and used for pattern recognition.

For our IAT-based counterfeit detection technique, we use an ANN-based classifier. The pattern recognition technique we use (patternnet function in MATLAB) involves the use of the *feedforward* class of ANNs on which to train and classify the IAT-based signatures. Feedforward ANNs are essentially acyclic variants of ANNs, with each neuron operating on the output of the neuron before it, thereby forming a complex one-way network of neuron functions. The feedforward ANN is set to use *scaled conjugate gradient backpropagation* as the training function. The feedforward network used has two layers, hidden and output. This kind of ANN can learn any input-output relationship provided there are enough hidden neurons. We empirically found 50 hidden neurons to be optimal for our application. For a traffic type, we form PDFs from IATs of blocks of 2,500 packets, and use them to train the ANN. This is done for each device configuration. The trained ANN then contains the signature database against which the test PDFs are compared. The ANN returns a value ranging from 0 (dissimilar) to 1 (identical) for each trained signature when fed a test PDF. The highest value from amongst these, which signifies the signature most similar to the test PDF, is chosen as the device to classify the test PDF. This training process is repeated for each traffic type, thereby creating a trained ANN for each traffic type. Then, the corresponding ANN is used for testing PDFs of each traffic type.

B. *Tests Conducted*

With the i7 processor, we captured all the traffic types mentioned. Then, we replaced the i7 processor on the motherboard with an Intel Core i3-3220 3.3 GHz processor (Figure 11) and repeated the experiments. For the captured traffic types, the PDFs of IATs were plotted. We found that ICMP requests (56 bytes) sent from the computer gave the best separation of PDFs. Figure 12 shows the PDFs of IATs of ICMP requests (56 bytes) sent out from the computer every 1 ms. The i3 PDFs peak at the required mean IAT of 1 ms (ICMP send rate), while the i7 PDFs have a smaller peak at 1 ms and two lateral peaks. This separation between the two can be differentiated by a classifier. Other traffic types are not so clearly separated. UDP results are almost the same except at the tips of the PDF peaks. Figure 13 shows the PDFs for UDP IATs.

In all, we had five i3s and five i7s. These IAT experiments were repeated with the four remaining i3s and i7s, and all traffic types were captured. For each traffic type, the following was done. The captured traffic's IATs were fed into an ANN-based classifier as PDFs with 300 bins formed from blocks of 2,500 packets. Histograms of an i3 and an i7 were set aside for testing, and the remaining four i3s and i7s were used to train the ANN. The training and testing were again repeated by choosing another i3 and i7 as the testing pair (with the remaining four i3s and i7s for training) each time, until all the i3 and i7 pairs had been used as the testing pair. Then, the five

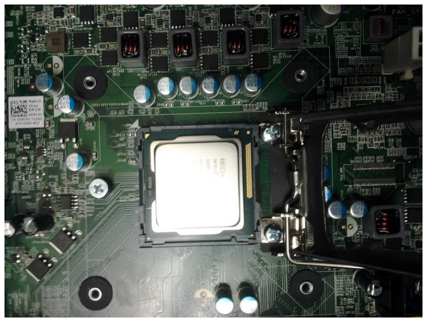

Fig. 11: Processor on the Optiplex 7010's motherboard.

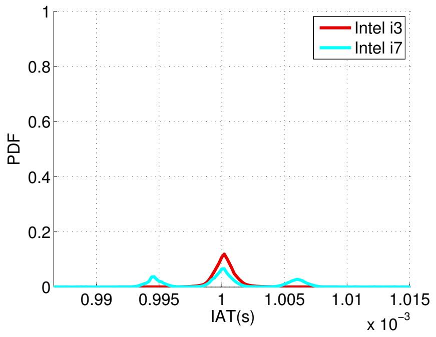

Fig. 12: PDFs of ICMP (56 bytes) request IATs for processor variations on Optiplex 7010.

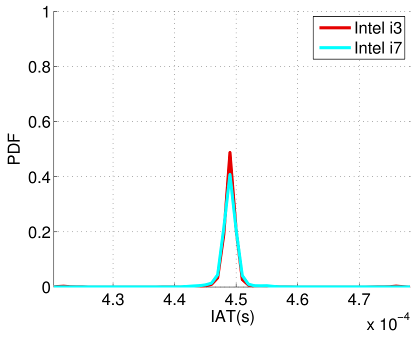

Fig. 13: PDFs of UDP IATs for processor variations on Optiplex 7010.

classifier results were averaged. The other traffic types were similarly used to train ANNs and run classification tests.

A good counterfeit detection technique is expected to detect as many counterfeits as possible while not making too many false detections. Hence, the recall value, which is the ratio of the number of true positives (Tp) to the sum of the number of true positives and false negatives (Fn), is an insightful measure of the actual effectiveness of a counterfeit detection technique.

$$Recall = \frac{Tp}{Tp + Fn} \tag{1}$$

Table I shows the recall values for the different traffic types for five i3s and i7s. The recall values for ICMP (56 bytes) requests are the highest at 0.78. Other traffic types have lower recall values of 0.61 to 0.66 due to less separation in the PDFs.

VI. Discussion and Limitations

Statistical analysis of IATs can be used to detect the modification of an internal component of a device like a processor. This can be done with a quick capture of just 2,500 packets to form IAT PDFs (which are used as signatures) that are compared to a signature database containing legitimate and illegitimate device signatures. In an implementation scenario, this legitimate signature could be provided by the manufacturer by a secure means to the customer, and the illegitimate signatures may be provided by a government agency (e.g., DHS or DIA). The recall results were shown for all traffic types, and the results for ICMP requests (56 bytes) traffic are the most promising for this technique of counterfeit detection. Other traffic types gave lower recall rates. The stability of the IAT signatures across multiple i3s and i7s was also shown.

Thus, this technique of counterfeit detection based on network traffic IATs is very fast, since it only needs 2,500 packets to provide results. It is cheap, since it does not need any expensive and specialized hardware to capture traffic and perform detection. The required equipment is a computer, nTAP, and NetFPGA. It is simple since it just involves capturing traffic and creating PDFs of IATs, hence there are no implementation difficulties. This method is non-invasive and non-destructive, as it monitors a device's network traffic without affecting the hardware or software of the device in any way. Thus, it can be used on devices without damaging them or corrupting their software. It is also, we believe, the first network-based counterfeit detection technique. Being that it is simple and network-based, it can be used on a wide range of devices and device types. Thus, it is also a broad-scoped technique, as all that is required is that the DUT has a network protocol stack and is able to send and receive traffic. This can be leveraged to deploy the technique remotely on a large network. This technique is also much faster and simpler than running benchmark software to determine changes to the device, as there is no need to individually access each device and install custom benchmarking software. The nodes can simply be booted, connected to a switch or access point, pinged (in the case of ICMP) for 2,500 packets, and classified.

However, the network-based nature of the technique brings with it the limitation that the DUT is required to be networked. This is a limitation that applies to all network-based approaches. Also, components that are non-crucial or those that do not directly affect the packet generation process will perhaps not be so easily detected by this technique. These component variations are perhaps more easily identifiable using a benchmarking technique. Another limitation of the technique is that when deployed across wide-area networks (WANs), it will end up fingerprinting the link itself, where the dynamic WAN paths introduce large delays and mask the subtle delays caused by the processors. Hence, this technique is only suitable for local-area networks (LANs).

VII. Conclusion and Future Work

This work illustrated the effects that device components have on network traffic. We characterized the individual effects that device components, such as a processor's clock, data cache, and instruction cache, have on different types of network traffic. This characterization was based on the IATs of the packets sent out of the FPGA DUT. The IATs were then visualized using PDFs. Each device component was varied one at a time, and the PDFs were compared to bring out the effects of that particular device component on network traffic.

We then used this technique of IAT-based analysis to determine the legitimacy of a device and its components. Counterfeit detection is of special interest because of all the counterfeit devices that have plagued industry and the military alike, costing them time, money, and putting lives at risk. Our technique is a simple and cheap, first-of-its-kind network-based approach to counterfeit detection. We conducted similar experiments as we did for the FPGAs on a real computer and obtained promising recall results of up to 78.7% using a classifier.

Possible future work includes, experimenting on more computers for different types of component combinations. We plan to test other processors like Intel Core i5. We will also run experiments that vary multiple components at a time to observe the effects they have on network traffic. Also, we plan to use other classification algorithms (e.g., cosine similarity, cross-

TABLE I: Recall values for five i3s and i7s for various traffic types.

	TCP	UDP	ICMP (56 B) response	ICMP (1400 B) response	ICMP (56 B) request	ICMP (1400 B) request
i7	0.776	0.613	0.674	0.698	0.746	0.743
i3	0.560	0.679	0.585	0.527	0.827	0.527
Traffic avg	0.668	0.646	0.630	0.613	0.787	0.635

correlation) to compare with the ANN-based one we have used. We will also examine the potency of the technique when targeting a loaded system. While we can expect a new system to be idle when we apply our counterfeit detection technique, it is useful if one could check the legitimacy of systems that are already deployed in networks and are executing jobs that cannot be halted for the counterfeit tests.

Acknowledgment

This work was partly supported by NSF-CAREER-CNS-0545667 844144 and DARPA-N10AP20022.

References

[1] C.-C. Tschang *et al.*, "Dangerous fakes," http://www.businessweek.com/stories/2008-10-01/dangerous-fakes.

[2] M. Pecht and S. Tiku, "Bogus: Electronic manufacturing and consumers confront a rising tide of counterfeit electronics," *IEEE Spectrum*, vol. 43, no. 5, pp. 37–46, 2006.

[3] A. Kimery, "Pitfalls of counterfeit-part epidemic exposed," http://www.hstoday.us/single-article/pitfalls-of-counterfeit-part-epidemic-exposed/12deeefeabb5d4b3790782be1d89e5c9.html.

[4] USDOC, "Defense industrial base assessment: Counterfeit electronics," http://www.bis.doc.gov/defenseindustrialbaseprograms/osies/defmarketresearchrpts/final_counterfeit_electronics_report.pdf.

[5] F. Koushanfar *et al.*, "Can EDA combat the rise of electronic counterfeiting?" *ACM/EDAC/IEEE Design Automation Conference, 2012*, pp. 133–138.

[6] Academy of Aerospace Quality, "Counterfeit parts tutorial: Effects of counterfeit parts," http://aaq.auburn.edu/node/135.

[7] U.S. Senate Committee on Armed Services, "Inquiry into counterfeit electronic parts in the Department of Defense supply chain," http://www.armed-services.senate.gov/Publications/Counterfeit%20Electronic%20Parts.pdf.

[8] H. Livingston, "Preventing and detecting counterfeit electronic components in the supply chain," http://www.nhjes.org/2011_Joint_Conference/presentations/2BNHJES_Livingston_Henry_Oct6%5B2%5D.pdf.

[9] J. Stradley and D. Karraker, "The electronic part supply chain and risks of counterfeit parts in defense applications," *IEEE Transactions on Components and Packaging Technologies*, vol. 29, no. 3, pp. 703–705, 2006.

[10] S. S. Hsu, "U.S. charges Florida pair with selling counterfeit computer chips from China to the U.S. Navy and military," http://www.washingtonpost.com/wp-dyn/content/article/2010/09/14/AR2010091406468.html.

[11] S. Pope, "Trusted integrated circuit strategy," *IEEE Transactions on Components and Packaging Technologies*, vol. 31, no. 1, pp. 230–234, 2008.

[12] W. Cobb *et al.*, "Physical layer identification of embedded devices using RF-DNA fingerprinting," *IEEE Military Communications Conference, 2010*, pp. 2168–2173.

[13] F. McFadden and R. Arnold, "Supply chain risk mitigation for IT electronics," *IEEE Conference on Technologies for Homeland Security, 2010*, pp. 49–55.

[14] K. Gao *et al.*, "A passive approach to wireless device fingerprinting," *IEEE/IFIP International Conference on Dependable Systems and Networks, 2010*, pp. 383–392.

[15] C. Neumann *et al.*, "An empirical study of passive 802.11 device fingerprinting," *IEEE International Conference on Distributed Computing Systems Workshops, 2012*, pp. 593–602.

[16] L. Watkins *et al.*, "A passive solution to the CPU resource discovery problem in cluster grid networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 12, pp. 2000–2007, 2011.

[17] L. Watkins *et al.*, "A passive solution to the memory resource discovery problem in computational clusters," *IEEE Transactions on Network and Service Management*, vol. 7, no. 4, pp. 218–230, 2010.

[18] L. Watkins *et al.*, "Using network traffic to remotely identify the type of applications executing on mobile devices," *IEEE Mobile Security Technologies*, 2013.

[19] B. Sieka, "Active fingerprinting of 802.11 devices by timing analysis," *IEEE Consumer Communications and Networking Conference, 2006*, pp. 15–19.

[20] S. Malhotra, "Fake Intel processors sold in Wazirpur IT market," http://www.dqweek.com/dq-week/news/16807/fake-intel-processors-sold-wazirpur-it-market.

[21] M. Singer, "Fake chips shadow AMD's new alchemy," http://www.internetnews.com/ent-news/article.php/3453541/Fake+Chips+Shadow+AMDs+New+Alchemy.htm.

[22] A. Shilov, "Attention! remarked AMD Athlon XP processors on the market," http://www.xbitlabs.com/news/cpu/display/20030503024652.html.

[23] M. Tehranipoor and C. Wang, *Introduction to hardware security and trust*, M. Tehranipoor and C. Wang, Eds. Springer, 2011.

[24] A. Kahng *et al.*, "Copy detection for intellectual property protection of VLSI designs," *IEEE/ACM International Conference on Computer-Aided Design, 1999*, pp. 600–604.

[25] F. Koushanfar and M. Potkonjak, "CAD-based security, cryptography, and digital rights management," *ACM/IEEE Design Automation Conference, 2007*, pp. 268–269.

[26] S. Wei *et al.*, "Gate-level characterization: Foundations and hardware security applications," *ACM/IEEE Design Automation Conference, 2010*, pp. 222–227.

[27] S. Wei *et al.*, "Robust passive hardware metering," *IEEE/ACM International Conference on Computer-Aided Design, 2011*, pp. 802–809.

[28] A. Caldwell *et al.*, "Effective iterative techniques for fingerprinting design IP," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 2, pp. 208–215, 2004.

[29] B. Gassend *et al.*, "Silicon physical random functions," *ACM Conference on Computer and Communications Security, 2002*, pp. 148–160.

[30] B. Gassend *et al.*, "Controlled physical random functions," *IEEE Computer Security Applications Conference, 2002*, pp. 149–160.

[31] M. Majzoobi *et al.*, "Techniques for design and implementation of secure reconfigurable PUFs," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 2, no. 1, 2009.

[32] T. Wolf and M. Franklin, "Commbench-a telecommunications benchmark for network processors," *IEEE International Symposium on Performance Analysis of Systems and Software, 2000*, pp. 154–162.

[33] C. Torrioni, "X-ray imaging to detect counterfeits," *SMT Magazine, Aug 2012*, pp. 44–47.

[34] W. R. Stevens, *TCPIP Illustrated, Volume 1- The Protocols*. Addison-Wesley Professional, 1993.

[35] *NetFPGA 1G*, http://netfpga.org/1G_specs.html.

[36] A. Babikyan, "Sharktools 0.1.5: Matshark and pyshark," http://www.mit.edu/~armenb/sharktools/.

[37] *GR-XC3S-1500 Development Board*, http://www.pender.ch/docs/GR-XC3S-1500_user_manual_rev2-0.pdf.

[38] *XtremeDSP Starter Platform*, http://www.xilinx.com/support/documentation/boards_and_kits/ug454_sp3a_dsp_start_ug.pdf.

[39] *LEON3 Processor*, http://gaisler.com/index.php/products/processors/leon3, Aeroflex Gaisler.

[40] *BusyBox*, http://www.busybox.net/.

[41] *GRMON Debugger*, http://www.gaisler.com/index.php/products/debug-tools/grmon.