




A SDR-based verification platform for 802.11 PHY layer security authentication

Xiaoguang Li¹ · Jun Liu¹  · Boyan Ding¹ · Zhiwei Li¹ · Haoyang Wu¹ · Tao Wang¹

Received: 3 October 2018 / Revised: 7 November 2018 / Accepted: 26 November 2018 /
Published online: 23 January 2019
© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract

The WiFi security authentication mechanism combined with the PHY layer information has become a hot spot of WiFi security research. The PHY layer contains rich information such as wireless channel, device location, and signal quality. High performance WiFi verification that supports PHY layer programming has become an indispensable tool for WiFi security research. This paper designs and implements a verification platform TickSEC that supports the research of WiFi security authentication at the PHY layer. It supports real-time acquisition of PHY layer information, and offers the programmability within the PHY layer. We also give a case study of WiFi device identification using PHY layer information. Experimental results show that TickSEC can meet the needs of PHY layer WiFi authentication verification.

Keywords Software defined radio · WiFi security · PHY layer · FPGA

This article belongs to the Topical Collection: *Special Issue on Security and Privacy in Network Computing*

Guest Editors: Xiaohong Jiang, Yongzhi Wang, Tarik Taleb, and Hua Wang

NaNA 2018 conference's recommendation paper

✉ Jun Liu
juneliu@pku.edu.cn

✉ Tao Wang
wangtao@pku.edu.cn

Xiaoguang Li
lixiaoguang@pku.edu.cn

Boyan Ding
dboyang@pku.edu.cn

Zhiwei Li
lizhiwei@pku.edu.cn

Haoyang Wu
wuhaoyang@pku.edu.cn

¹ Peking University, Beijing, China

1 Introduction

In modern society, the popularity of WiFi facilitates people's life. In the meantime, it has also brought security risks. The security mechanism of the PHY layer has become a hot topic in WiFi security research. The PHY layer contains rich information on wireless channels, such as equipment locations and signal quality, and researchers have sought to use these to enhance the security of WiFi. Examples of information that researchers have used include RSS (Received Signal Strength) [37], CIR (Channel Impulse Response) [20, 21], CSI (Channel State Information) [22] and so on.

WiFi security research requires a verification platform to implement and evaluate security mechanism. Compared with the security research of the upper layers (e.g. TCP/IP), PHY layer security research places higher requirements on the verification platform. The PHY layer data transmission rate specified in the 802.11ac protocol is 433Mbps (single antenna, 80MHz bandwidth), with PHY layer delay not exceeding several microseconds. PHY layer implemented with software cannot meet the requirements in terms of throughput and latency. While hardware implementations are often not programmable enough to fit the needs of security research.

This paper mainly has the following contributions:

- Though analysis of common WiFi attack and three strategies of using the PHY layer information to deal with WiFi attack, we choose the PHY layer information (such as CSI, RSSI and frequency offset) for security authentication;
- Based on previous work Tick [31], we design and implement TickSEC (Tick for Security) platform that supports PHY layer security authentication. Different from commercial network cards, it can obtain PHY layer information in real time while providing programmability, convenient for PHY layer security verification.
- As a case study, an approach for identifying different WiFi devices using PHY layer information is proposed. We verify the effectiveness of TickSEC and introduce a machine learning method to analyze the results of security authentication.

The remainder of this paper is organized as follows. Section 2 introduces the PHY layer of 802.11 and PHY Layer Security Research. Section 3 presents related work in WiFi security research and several state-of-the-art SDR platforms for WiFi research. Sections 4 and 5 analyze the design and implementation of TickSEC respectively, highlighting its support for WiFi security research at PHY layer. Section 6 proposes a case study and the evaluation of platform. Section 7 is the summary this paper.

2 Background: The 802.11 PHY layer

The 802.11 PHY layer specifies the mode of modulation, demodulation, coding, and radio frequency parameters of data transmission and constitutes the lowest layer of the OSI network model. Except the legacy 802.11b, which uses direct-sequence spread spectrum (DSSS) modulation techniques, other 802.11 standards, i.e. 802.11a/g/n/ac, use OFDM (orthogonal frequency-division multiplexing) as their modulation technique. So our work focuses on the more contemporary OFDM.

2.1 The 802.11 processing pipeline

Figure 1 depicts a block diagram of the 802.11 PHY layer processing pipeline.

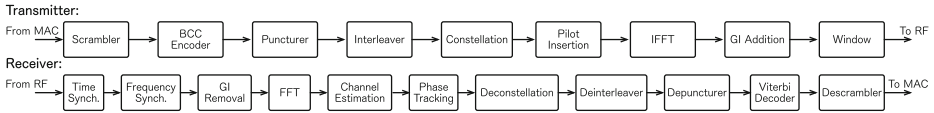


Figure 1 Transmitter and receiver modules for 802.11 a/g/n/ac

On the transmitter end, the PHY layer obtains a frame of binary data from the MAC layer. The frame data passes through source coding, constellation mapping and IFFT (Inverse Fast Fourier Transform). Then guard intervals (GI) are inserted in each symbol, and a preamble is prepended before the frame. Finally the data samples are sent to radio frequency (RF) frontend reaches the air.

The receiving end, on the other hand, converts RF signal received in the air back into frame data. The PHY first acquires wireless signal from the air in the form of I/Q complex samples from the RF frontend. The samples are tracked in the time synchronization modules, which determines the start of a frame. When time synchronization succeeds, the samples are passed through the following modules, including frequency synchronization, GI removal and FFT, which converts time domain signal back into frequency domain. After FFT, the channel estimation module does equalization, removing the influence of the channel on the signal. Then the frame data is solved with a process that is a inverse of the transmitter.

The 802.11 PHY layer contains a considerable amount of valuable information for WiFi security research, especially at the receiving side. CSI can be obtained from the channel estimation module, reflecting the location of the transmitter, the surrounding environment, etc. Other information, such as RSSI, frequency offset [29], error vector magnitude (EVM) [28], can be acquired from time synchronization, frequency synchronization and constellation modules respectively.

2.2 Information from 802.11 PHY layer

This section describes some PHY layer information that is common in WiFi security research.

RSSI (Received Signal Strength Indicator) is a measurement of the power present in a received radio signal, related to wireless transmitter power, transmission and reception distance, and surrounding obstacle environment. Researchers often use RSSI to distinguish between different devices or to obtain changes in the surrounding environment.

CSI (Channel State Information) refers to known channel properties of a communication link. This information describes how a signal propagates from the transmitter to the receiver and represents the combined effect of, for example, scattering, fading, and power decay with distance. CSI has a wealth of physical ubiquity that is often used to extrapolate other information, such as location and movement.

Frequency offset is an offset between the carrier frequency of the received signal and the transmitted signal. In radio engineering, a frequency offset is an intentional slight shift of broadcast radio frequency (RF), to reduce interference with other transmitters. The frequency offset is estimated based on the offset of the period, and then the frequency offset compensation is performed.

3 Related work

3.1 WiFi PHY layer security research

3.1.1 Hardware fingerprint

Hardware fingerprint refers to the technology of identifying and authenticating hardware-related PHY layer information. The hardware related PHY layer information in WiFi include clock offset, clock fluctuation, and radio frequency characteristics. The PARADIS [8] system uses a variety of PHY layer information combined with the machine learning model to identify 802.11 wireless devices. This paper collects PHY layer information related to the hardware, including frequency offset and offset of constellation points. The experiments in this article show that the PARADIS system can identify more than 130 commercial NICs with an accuracy of over 99%. There are also other wireless systems, such as Bluetooth, which use hardware fingerprint technology for identification. For example, the BlueID [13] system uses clock characteristics of Bluetooth devices for device identification.

3.1.2 Channel fingerprint

Channel fingerprint refers to the technology of identifying and authenticating by using channel-related PHY layer information. The channel-related phy-layer information is often used to locate and judge whether the device is moved. The information includes CSI, RSSI, multi-antenna signal direction matrix, et.al. Faria [9] used RSSI to authenticate wireless devices. Bagci [6] suggested using CSI to detect whether the Internet of Devices is unlawful. This article uses the change of CSI to judge whether the device is moved. Xiong [35] used CSI to recognize a variety of active attacks. Multi-day line technology is used in the 802.11n protocol to significantly increase system transfer rate and accuracy. This article uses the multi-antenna CSI to extract the signal to the angle of view, and proposes a set of DataCheck protocols that are certified by the angle of arrival to identify and authenticate the device. It is possible to identify devices that are 5 cm apart, and the recognition accuracy is significantly improved compared to the 5 m by using RSSI.

The physical layer security mechanism is based on the security mechanism of the encryption technology and the non-encrypted, also known as the authentication-based security mechanisms. The security mechanism based on encryption has certain limitations. Complex encryption algorithms are difficult to implement in the actual communication system [24]. Therefore, some researchers explore the authentication-based security mechanisms [37]. Instead of encrypting data, works in this approach leverage PHY layer information to check the authenticity of wireless frames and devices. Table 1 provides a summary of existing works in this approach. According to the table, these systems have the following requirements for an ideal verification platform:

- Provides common PHY layer information, such as CSI, RSSI, etc., in addition to supporting extended PHY layer information like frequency offset, signal correlation [11], constellation point offset [12] and CIR [20].
- Customize the PHY layer data processing. For example, research has introduced artificial frequency offsets at the PHY layer sending end [23]. This can only be done by platforms that support PHY layer programming.
- Communicates in real time with commercial devices, increasing the credibility of verification results.

Table 1 Summary of authentication systems based on PHY layer information

Existing works	PHY info	Verification platform
TPDS'13 [36]	RSS	Orinoco silvercard, Atheros miniPCI NIC
TVT'16 [34]	RSSI	USRPN210
MobiSys'10 [16]	RSS	Nokia N800 Internet tablets
MILCOM'11 [20]	CIR	Simulation
ICC'13 [21]	CIR	Simulation
INFOCOM'13 [22]	CSI	Intel IWL5300 NIC
AISA CCS'14 [23]	Freq. offset	USRPN
ICC'07 [33]	Freq. offset	Wise [10]

- Support process modularity of PHY layer. The processing of the PHY layer information requires intensive calculations, and the software part can quickly implement the encryption algorithm while the hardware part can improve the efficiency and meet the timing requirements of the protocol.

3.2 WiFi research platforms

The existing wireless platforms that support WiFi research at the PHY layer are mainly divided into four types:

1. Computer simulation softwares without RF front end. On these platforms, PHY layer implemented with pure software for simulation. Notable examples of this type are Matlab [27] and Wise [10]. The former provides a programming language suitable for scientific computing and a rich collection of tools for wireless algorithms development, supporting a number of WiFi researches. While the latter contains modeling and simulation tools specific to wireless communication.
2. Commercial network cards with RF front-end and PHY layer implemented on ASIC. These platforms provide ideal performance for real-time communication and verification. However their programming capability is limited by their proprietary and fixed-function implementation on hardware. Intel IWL5300 is one of the common NICs used by various studies [22, 32].
3. Software radio platform, with RF front-end, PHY layer implemented by software. A famous representation of this type is the combination of National Instruments hardware USRP and the open-source software GNU Radio [7]. GNU Radio allows users to develop their own PHY layer modules in software and also provides a graphical development environment to facilitate the customization of data processing flow. Compared with pure software simulation, it is advantageous in its ability to communicate in the real wireless environment. However, the performance limitation of software makes it impossible to implement high-throughput protocols such as 802.11 in real time.
4. FPGA-based platforms with RF front-end and PHY layer implemented by FPGA hardware. FPGA offers both the high performance of hardware and programmability through hardware description languages (HDL), ideal for acceleration of customized applications. Several platforms have been developed these years in the field of wireless research. Rice University's WARP platform [17] is an example.

These four types of wireless verification platforms have different characteristics and advantages. However, currently none of the platform is ideal for PHY layer security research. Computer simulation software has the best programmability and a large amount of public code for research, but it cannot be verified in a real wireless environment. The performance of commercial network cards is the best and real environment verification is possible, but the programmability is the worst in the four types. Software-based radio platforms have high PHY layer programmability and can do wireless communication, but their communication performance is low and they cannot meet the requirements of the WiFi protocol standard. FPGA-based radio platforms have the best potential, with both high performance and reasonable programmability. However, these platforms are still in early stages in terms of maturity and they are not optimized for security research. Table 2 provides a comparison of these types of platforms.

Therefore, researchers at the PHY layer for WiFi security need a wireless open platform that satisfies their needs in both performance and programmability to implement and validate their research.

Our work is based upon the Tick [31] platform, which provides a high-performance and programmable SDR implementation that supports 802.11a/g/ac protocol. The original Tick isn't optimized for security research, where [18] proposes some preliminary ideas on the engineering aspect. We further develop upon the ideas of the latter work and formed a architecture design and implementation. Li [18] contains more technological details of certain modules described here.

4 Design of TickSEC

4.1 Design goals

Based on the above requirements on verification platforms for authentication, the following design goals of the verification platform should be met:

1. Realize 802.11 protocol, enabling real-time communication with commercial wireless network card;
2. Support customization of data processing in PHY layer;

Table 2 Comparison of different types of WiFi research platforms

Platform (example)	PHY info extraction	PHY layer programmability	Interoperability	Performance
Simulation SW (Matlab)	No	Good	No	Poor
Wireless NIC (Intel IWL5300)	Partly	No	Yes	Good
SW-based SDR (GNU Radio)	Yes	Good	Not real time	Poor
FPGA-based SDR (WARP)	Yes	Poor	Yes	Good

3. Easy access to commonly-used PHY layer information, such as RSSI, CSI, frequency offset, etc., and support user extension to acquire other PHY layer information;
4. Support software and hardware implementations of security algorithms, providing the ability to interchange between hardware and software modules.

4.2 Design challenges

the design challenge TickSEC face are as below:

1. PHY layer information extraction. We need to obtain various information in PHY layers while ensuring high performance. How to obtain the spectrum information from PHY layer in real-time without affecting normal communication is a challenge. We also need to solve the problems in analysing spectrum conditions from these information.
2. Low-latency and high throughput transmission. The extraction and transmission latency of PHY layer is very important for WiFi security applications. How to achieve low latency and high throughput transmission of PHY layer information is a challenge.
3. One verification platform should not only satisfy the existing research, but also provide a certain degree of extensibility to support future innovations. How to design the interface to improve the scalability is another challenge.

4.3 TickSEC architecture design

To overcome the above-mentioned challenges, we propose the corresponding architectural designs of TickSEC based upon the original Tick [31] system. The architecture of TickSEC extends in three aspects: First, the scope of software and hardware collaboration is extended, and the PHY layer is also connected to the embedded processor through the AXI bus, so that users can directly configure the PHY layer and to obtain data from the it; second, TickSEC provides users with software and hardware programming interface to extract the commonly used PHY layer information; third, extensible hardware module to analyze the PHY layer information and its interface with software is designed. The advantage of this approach is that WiFi security researchers can easily obtain common PHY layer information from the software side and expand other PHY layer information, and can choose to implement their own security mechanisms by hardware or software for verification.

4.3.1 Overview of TickSEC

The hardware structure of TickSEC is illustrated in Figure 2. The block design in the figure is the part of the software and hardware co-design. The USB communication library used to provide connection with the host PC is placed outside the block design. Core module in block design is an embedded MicroBlaze processor. Low Mac TX and RX are the sender and receiver parts of the Low MAC, which is responsible of the timing-critical logic of the MAC layers, and are reserved from Tick design. In TickSEC, we package PHY TX and RX modules, which were outside in Tick, into IPs and place them into block design. This enables their connection with the embedded processor through the AXI bus, and the radio frequency (RF) support library also moves into the block design. The PHY Analyzer mode newly added hardware IP of TickSEC. It is used to perform data analysis on the PHY layer information obtained from the PHY layer module and feed the analysis result back to the embedded processor.

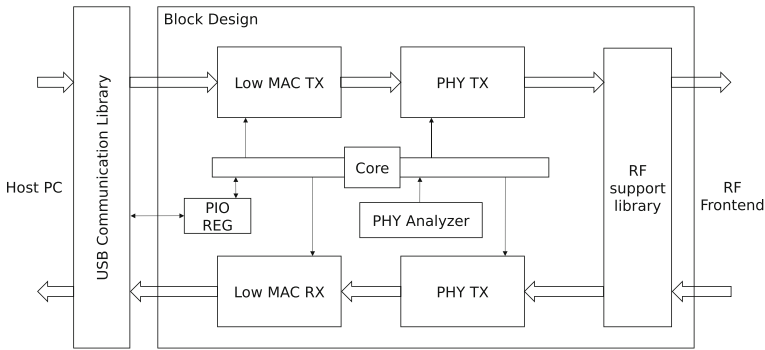


Figure 2 TickSEC hardware module design

4.3.2 Extraction of common PHY layer information

In this part, several commonly used PHY layer information extraction methods with frames are described, specifically including CSI, RSSI, frequency offset and clock offset. In above, we mentioned that CSI and RSSI are the most commonly used PHY layer information in security research. Frequency offsets are often used in authentication-based security mechanisms. For the AP, the protocol specifies that a beacon frame is transmitted every 100ms. We can obtain the beacon frame by receiving the timestamp of the beacon frame thus clock offset can be calculated from the beacons. The following will introduce the extraction process of CSI, RSSI and frequency offset respectively.

Before discussing ways to extract CSI and other PHY layer information, we'll first present a brief introduction of 802.11 frame, especially fields in the preamble to facilitate understanding of the extraction algorithms.

Figure 3 illustrates the structure of the 802.11a/g frame at the PHY layer. The preamble at the beginning of the frame is prepended by the sender before the data and is known to both parties. The receiver performs synchronization and channel estimation based on preamble. The preamble in 802.11a/g PHY layer contains two training fields. The first is short training sequence (STS) of 160 sampling points, followed by long training sequence (LTS) of another 160 sampling points. SIGNAL field (80 sampling points) comes after these training fields, specifying the frame length and modulation method. The short training word takes 16 sampling points as a cycle, a total of 10 cycles, sampling interval is $0.05\mu s$, used to do AGC (automatic gain control), time synchronization, frequency synchronization, etc., will be described later when extracting frequency deviation and RSSI. The first 32 sampling points of the long training word are guard intervals, which are composed of 16 sampling

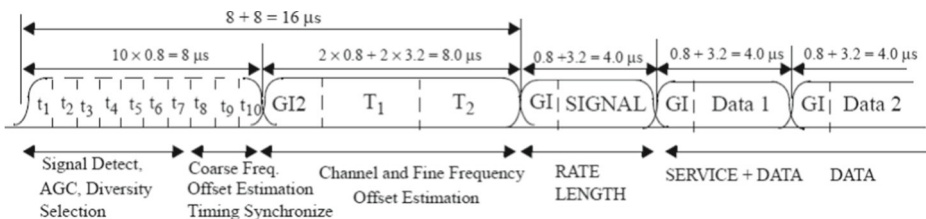


Figure 3 802.11a/g Frame structure

points T_1 and T_2 . T_1 and T_2 are repeated. The long training words are used for channel estimation and fine-grained frequency offset estimation.

CSI The CSI of 802.11 refers to the current channel status of each subcarrier. It is a common assumption that the CSI doesn't change substantially within the reception process of an 802.11 frame. Based on this assumption, the 802.11 implementations often uses the Long Training Sequence (LTS) in the preamble to estimate the channel. The extraction CSI is done within the channel estimation module which analyses the LTS.

The frequency domain values of the LTS is a sequence of 1 and -1 s, as show in Figure 4, without other values (the 0 in the middle is the DC component which doesn't carry information). So to obtain CSI from LTS, we only need to convert the sign of each subcarrier according to Figure 4. On the other hand, the STS is not suitable for CSI calculation because it also contains other values such as 0 and $1 + j$. In the specific implementation of TickSEC, we prestore the theoretical value of the long training word and compare it with the long training word received. When the theoretical value is -1 , the sign of received value inverted, and the theoretical value is 1 when it is unchanged. After the sign conversion, a vector of 52 samples is obtained. This vector can represent the CSI of this frame.

RSSI The software driver of the Tick RF communication library provides a function to directly calculate the RSSI. However, in practical tests, it is found that the execution of this function is relatively slow, lasting several milliseconds, so the result does not represent the RSSI of the current frame. In order to extract the RSSI more accurately, TickSEC calculates the power sum of the short training words in the synchronization module, because the theoretical power value of the short training words of different frames is fixed, and unlike the long training words, which has a large subcarrier span. However, the automatic gain control (AGC) at the RF frontend will affect the value received at PHY layer. Thus the gain at AGC, can be read from RF frontend with low latency, should also be taken into account in RSSI calculation.

Frequency offset The frequency offset is extracted in the frequency offset estimation module and utilizes the periodicity of the STS, specifically, the last 64 sampling points (4 out of 10 cycles) of the short training word is used. There are two reasons for this partial utilization. First, the previous sample points are affected by AGC mentioned above, and their amplitude fluctuates. Using them might degrade the accuracy of frequency offset calculation. Second, the frame is usually synchronized at the middle of STS, and the front beginning may not be synchronized. Experiments show that taking the last 4 cycles is a feasible solution.

We first transform the frequency offset equations. Assuming that the received values at times k_1 and k_2 are $r(k_1)$ and $r(k_2)$ and the theoretical values are $s(k_1)$ and $s(k_2)$, there are the following equations:

$$r(k_1) = s(k_1) \cdot e^{j2\pi \Delta f k_1 / f_s} \quad (1)$$

$$r(k_2) = s(k_2) \cdot e^{j2\pi \Delta f k_2 / f_s} \quad (2)$$

$L_{-26, 26} = \{1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 0,$
 $1, -1, -1, 1, 1, -1, 1, -1, 1, -1, -1, -1, -1, -1, 1, 1, -1, -1, 1, -1, 1, 1, 1, 1\}$

Figure 4 Frequency domain values of 802.11a/g LTS

Among them, Δf is the frequency offset, f_s is the sampling frequency, the value of which is 20M in 802.11a/g. $r(k_1)$, $r(k_2)$, $s(k_1)$, $s(k_2)$, and f_s are all known. We can find Δf using the following method. According to the characteristics of the short training word, i.e., its 16 sampling points repetition, when $k_2 = k_1 + 16$, $s(k_1) = s(k_2)$, substituting the values the previous equation, we can get the following:

$$\text{Let } \frac{r(k_1)}{r(k_2)} = A + Bj \quad (3)$$

$$\Delta f = \frac{f_s}{2\pi \cdot 16} \arctan\left(\frac{B}{A}\right) \quad (4)$$

Equation (3) is derived from the properties of complex numbers. $r(k_1)$ and $r(k_2)$ are complex numbers, so the result of division is also a complex number. The complex number can always be written in the form of $A + Bj$, where A and B are real values. Equation (4) is obtained by substituting (3) into (1) and (2) to obtain the frequency offset value.

In the implementation of TickSEC, we get A and B in the hardware, these values are passed to the software. Then by the software calculates \arctan and multiplication to get frequency deviation. When hardware calculations A and B are needed, every 16 receivers Dividing the sample points and dividing the complex number $r(k_1)/r(k_2)$ into $r(k_1)r(k_2)^*/|r(k_2)|^2$.

We get a set of A and B from the 16th sampling point before it. We add 64 A and B obtained from 64 sampling points to average, reduce the error, and finally the extracted PHY layer information is the average A . B , denotes the periodic shift real part and the periodic shift imaginary part, respectively.

4.3.3 Alignment of the PHY layer information

After the design of the extraction of PHY layer information, we face two practical problems: one is the frame alignment problem of the PHY layer information, and the other is the exact reception time problem. The frame alignment problem of the PHY layer information means that when the software MAC layer receives a frame, the PHY layer information corresponding to the frame is taken from the PHY layer, not the information of previous frame or next frame. This problems arises from the fact that the MAC and PHY layers in Tick system works asynchronously. So how we can make the MAC layer get the correct PHY layer information without blocking the PHY layer has become a problem. The exact reception time problem refers to how the software MAC layer acquires the precise arrival time of the frame when it receives a frame. Jana and Kaseru [14] proposes to use the frame arrival time as the basis for calculating the hardware fingerprint, and the arrival time must be accurate, i.e., in μs level. However, the asynchronous nature of MAC-PHY interaction also makes the time stamp at MAC layer inaccurate. Our solution of these two problems are discussed in the following.

For the alignment of the PHY layer information, we use a three-level cache illustrated in Figure 5. When the PHY layer receives a frame, each module calculates the PHY layer information through the preamble. The information is first stored inside the module and is aligned with the clock of the module. This is called intra-module buffering. When the SIGNAL field, which follows the preamble of the frame, is correctly decoded and the Low MAC is ready to receive the current frame, the PHY layer notifies the Low MAC module that it has received a frame and initiates an interrupt to the embedded processor. At this time, TickSEC will cache the PHY layer information from various modules in another set of unified clock registers through clock domain conversion. The unified module is known

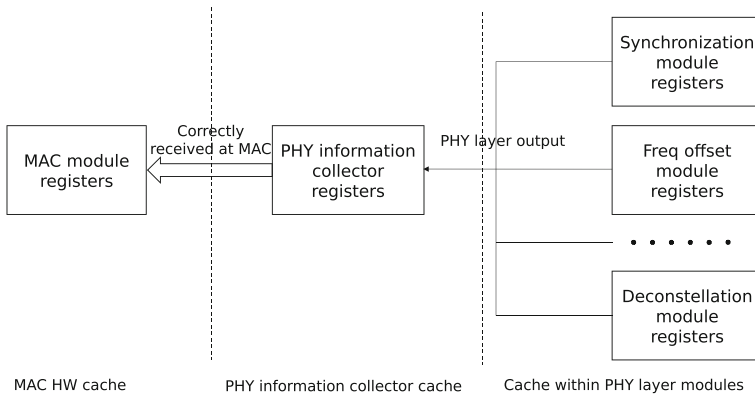


Figure 5 Three-level buffering framework

as the PHY layer information collection module cache. The purpose of this step is to allow each module to continue processing the next frame and update the in-module cache.

After the Low MAC software receives the interrupt, the PHY layer information cached by the PHY layer information collection module is stored in the third group of registers aligned with the AXI bus clock, called the MAC layer hardware cache, and the Low MAC software reads the data through the AXI bus. The PHY layer information cached by the MAC layer hardware reads the PHY layer information of the frame that received the interrupt. The purpose of this step is to allow the PHY layer to initiate the next frame of the interrupt to the Low MAC. The processor can read the PHY layer information of this frame at any later position within a frame. The advantage of adopting the three-level buffering method is that while ensuring that the data segment of one frame and the PHY layer information keep synchronizing, the reception and data processing of the following frame are not blocked.

Another possible solution of the alignment problem is using a FIFO queue to store PHY layer information. But we find this solution infeasible because the PHY layer will discard frames under certain conditions, such as unrecognized modulation method, too long frame length, and embedded software may also. The interrupt will be lost. Once the PHY layer drops the frame or the interrupt is not processed, the buffered PHY layer information will not be read. The queue is blocked and information gets dislocated. This situation is depicted in Figure 6. If buffering is in the register, the PHY layer information of the following frame will cover the previous one when an error occurs, and will not cause misalignment.

4.3.4 Design of accurate reception time

For accurate reception problems, TickSEC uses the PHY layer time stamp method. First, we use a 64-bit counter at the PHY layer and expose it to the Low MAC interface. At each clock cycle the counter is incremented by one. The clock frequency of this module here is 100MHz, so the precision of the counter is 10ns. Then, when the PHY layer synchronizes to a frame, the counter value is read and held. The Low MAC software will read the saved counter value through the AXI bus when it receives an interrupt from PHY layer. This method uses the value of the counter as a type of PHY layer information and uses a three-level buffering technique for recording. The counter is equivalent to buffering in the module. AXI bus read and write width is 32 bits. In general, the low 32-bit can record about 43 seconds time interval. For more than 43 seconds, the high 32-bit should be used.

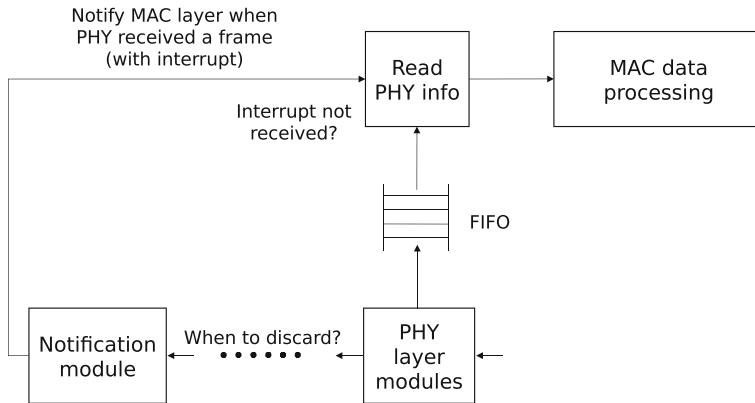


Figure 6 Dislocation problem when using FIFO

The advantage of using the PHY layer time stamp is that the user can accurately learn the relative reception time of the current frame. On the one hand, it can calculate the clock offset as a PHY layer information alone [14]. On the other hand, it can be combined with other PHY layer information. The relationship between other PHY layer information over time and the time-varying PHY layer information also have important physical meanings.

5 Implementation

5.1 PHY layer software interface

PHY layer software interface provide a way to extract PHY layer information from hardware modules into software. The TickSEC software code contains embedded software part on Microblaze of Low MAC and host driver part on PC. The software interface should support data acquisition on both parts.

We adopt a two-level programming structure of embedded software and host software. It first extracts the PHY layer information into the embedded software. Then the PHY layer information is transferred to the host via a PCIe/USB communication framework, instead of directly sending PHY layer information to the host driver. This structure has the following advantages:

- The Low MAC control flow is implemented by embedded software. The user may use PHY layer information directly in the Low MAC, for example, to determine whether to reply an ACK frame according to the PHY layer information.
- The embedded software includes an interface with the host PC. The PHY layer information is first extracted into the embedded software. The user can decide whether to continue uploading the information and maintain the centralized control process.
- The interaction between hardware logic and embedded software facilitates extension. If users want other information, they can simply add more AXI registers. However, if they are directly exported to the driver, the hardware logic of the USB communication library needs to be modified when extension is requires, which would require excessive changes to the communication library.

The specific implementation is described as follows. TickSEC hardware and software transfer values through AXI registers. Each register is 32 bits in width. Table 3 contains the definition of the AXI registers. These data are provided to both the embedded software on Microblaze for analysis in the Low MAC layer or on the host PC.

5.2 PHY layer information analysis module

TickSEC provides a hardware data analysis module for data analysis of PHY layer information to complement software analysis. The hardware data analysis module connected to the PHY layer to obtain PHY layer information and perform data processing. The embedded software processes data through the AXI interface.

This module process configures parameters and obtains processed data. The advantages of the hardware analysis module are high efficiency and strong real-time performance. In the hardware analysis module, we provide 32 sets of AXI registers, allowing users to interact with software and hardware.

5.3 PHY layer information extension

TickSEC also provides methods for users to expand PHY layer information in addition to common information such as CSI, frequency offset, RSSI, and modulation methods. Coupled with the fully programmable nature of Tick's own PHY layer, TickSEC can be well adapted to the increasingly evolving and changing needs of researchers.

The following describes an extension of PHY layer information with an example. In addition to frequency offset, [8] uses SYNC correlation and I/Q origin offset to generate

Table 3 Register definition of the TickSEC software programming interface

Reg ID	R/W	Register definition
0x00	W	Request to read PHY info
0x01	R	PHY info ready
0x02	R	Real part of Periodical offset, high 32 bits
0x03	R	Real part of Periodical offset, low 32 bits
0x04	R	Imag part of Periodical offset, high 32 bits
0x05	R	Imag part of Periodical offset, low 32 bits
0x06	R	Numerator of autocorrelation in synchronization
0x07	R	Autocorrelation normalization factor
0x08	R	Numerator of correlation with STS
0x09	R	Numerator of correlation with LTS
0x0A	R	Correlation normalization factor
0x0B	R	SIGNAL, containing frame length and modulation
0x0C	R	CSI of subcarrier -21
0x0D	R	CSI of subcarrier -7
0x0E	R	CSI of subcarrier +7
0x0F	R	CSI of subcarrier +21
0x10	R	High 32bits of timestamp
0x11	R	Low 32bits of timestamp

device fingerprints. We will next use synchronization correlation as an example to introduce how users can extend PHY layer information.

We'll first introduce the background of synchronization correlation. We use the STS for time synchronization, and use the periodicity of the STS which repeats every 16 sampling points. When the correlation (autocorrelation) between the adjacent 16 sampling points exceeds a certain threshold, we think that we have discovered the STS. This process is called frame synchronization, shown in Figure 7.

To calculate correlation, we use two adjacent sliding windows W_1 and W_2 . The sizes of both two windows are 16 sampling points. However, only frame synchronizing is not enough, because there are 10 repetitions of the short training words. We cannot determine which of the two adjacent groups are synchronized. To accurately locate the sampling points, LTS is also needed. We use LTS correlations to compare 16 sampling points with long training words. When the correlation (cross correlation) with long training words exceeds a certain threshold, and the autocorrelation is lower than the threshold at the same time, it is considered that the beginning sample of LTS is found. This time the beginning sample of LTS and frame data can be accurate synchronize. This process is called symbol synchronization and is shown in Figure 8.

When the sampling points in W_1 are all located in the STS, the sampling points in W_2 are all located in the LTS, the cross-correlation between the theoretical values of the first 16 points in LTS and W_2 will be above a certain threshold, and the correlation between points W_1 and W_2 will be lower than the threshold. Synchronizing dependencies include autocorrelation and cross-correlation. Brik et al. [8] pointed out that synchronous cross-correlation is related to hardware and can be used to identify devices.

The following uses cross-correlation as an example to introduce how to add a software interface with synchronizing dependencies.

The first step is to list the equations of synchronous cross-correlation, where P_n is a normalization factor.

$$C_n = \sum_{k=0}^{L-1} r_{n+k} \cdot s_{n+k}^* \quad (5)$$

$$P_n = \sum_{k=0}^{L-1} |r_{n+k}|^2 \cdot \sum_{k=0}^{L-1} |s_{n+k}|^2 \quad (6)$$

$$M_n = \frac{|C_n|^2}{P_n} \quad (7)$$

We know that FPGA hardware is more suitable for multiplication and addition than division. Since division in FPGA requires more clock cycles and has higher delay. Therefore, we can calculate C_n and P_n by hardware. In addition, because energy value of the long training words $\sum_{k=0}^{L-1} |s_{n+k}|^2$ is fixed, called P' , and we only need to calculate $\sum_{k=0}^{L-1} |r_{n+k}|^2$ in hardware. The calculation of M_n is done by software.

In the second step, C_n and P'_n are calculated in the synchronization module and are derived from the synchronization module. Add the port of the synchronization module, as

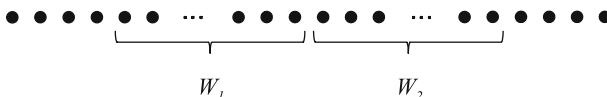


Figure 7 Sliding window of frame synchronization

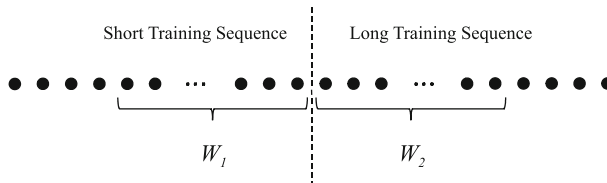


Figure 8 Sliding window of symbol synchronization

shown below. Here is the schematic code. Actually, in addition to C_n and P'_n , we have also derived several signals for calculating autocorrelation.

```

module rx_synchronization(
... // other ports
input signal valid,
output [31:0] sync_para_C,
output [31:0] sync_para_P1,
... // other ports
);

```

The `signal_valid` signal indicates that the subsequent module has successfully decoded the signal field. This value is 1 only if the signal field is valid, otherwise it is 0. We assign `sync_para_C` and `sync_para_P1` when `signal_valid` is 1.

The third step is clock domain conversion. This step converts `sync_para_C` and `sync_para_P1` in the corresponding clock domain of the synchronization module into `axi_sync_para_C` and `axi_sync_para_P1` in the AXI bus clock domain, and assigns them into AXI registers. TickSEC provides a clock domain conversion module `reg_clk_domain_switch`. Its usage here is listed as follows.

```

Reg_clk_domain_switch_64
SEC_SYNC_clk_switch_clk0_to_clkaxi(
  .clk_pre(usr_clk_ch0),
  .clk_pos(S_AXI_ACLK),
  .rst_pre(usr_rst_ch0),
  .rst_pos(S_AXI_ARESETN),
  .reg_pre(sync_para),
  .reg_pos(axi_sync_para)
);

```

The fourth step is to read C_n and P'_n from the embedded software code and use software to calculate the M_n we need. The embedded software code that reads and computes M_n is listed below. `SYNC_PARA_LTS` is the energy value of the pre-saved long training words.

```

int syncParaC = Xil_In32(PHYSyncPara1);
int syncParaP1 = Xil_In32(PHYSyncPara2);
float syncParaM = syncParaC * syncParaC /
  (syncParaP1 * SYNC_PARA_LTS);

```

At this point, using synchronous correlation as an example, the user has completed the expansion of the PHY layer information.

5.4 Multi-RF mode

In order to support the perfect channel, we have extended the RF communication library to provide two modes, RF mode and loopback mode. The RF mode provides functionality identical to the original RF communication library and the PHY layer is connected to the RF frontend normally. In loopback mode, on the other hand, the output of PHY transmitter is directly sent back to the receiver.

This approach facilitates debugging for WiFi security researchers in their early stage of system implementation. Using the loopback mode to simulate the perfect channel, researchers do not need to deal with the complexity the RF frontend. They can also obtain some result without the need to communicate between multiple devices. Also, the switch between the two modes is quite easy, the interfaces are compatible in the two modes and only one parameter needs to be change.

The following describes the specific implementation of the loopback mode, as shown in Figure 9. The data interface between the RF communication library and the PHY layer contains two asynchronous FIFOs. In the loopback mode, we directly connect the output of the sending FIFO to the input of the receiving FIFO. When there isn't data, blank signal is filled into the receiving FIFO to simulate blank time when there is no signal in the air. When the transmitter of the PHY layer notifies the radio frequency support library to send a frame, it enters the loopback mode and fetches data from the sending FIFO directly into the receiver FIFO.

It is worth noting that in addition to multiplexing of data streams, clock and reset signals must be multiplexed. The transmit and receive FIFOs are asynchronous FIFOs. In RF mode, the RF clock and reset signals are from the RF module. In loopback mode, they need to be changed to PHY layer clocking and reset signals, as shown below.

```
assign clk = (MODE == RFDMODELOOPBACK) ?
    PHY_clk : rf_clk;
assign rst = (MODE == RFDMODELOOPBACK) ?
    PHY_rst : rf_rst;
```

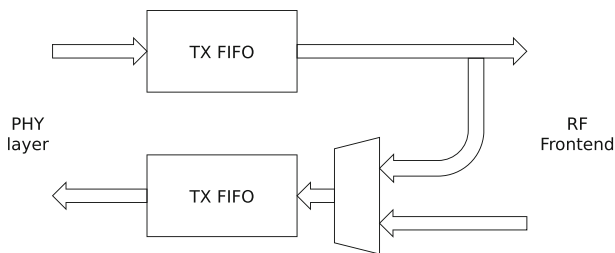


Figure 9 RF mode and loopback mode

6 Case study and evaluation

6.1 Case study setup

We give a case study for identify different devices in two implementation ways using frequency offset. For the software implementation of the platform, we use the first half of the off-line data to train the machine learning model, and the second half of the off-line record as the test set. We choose k-Nearest Neighbor (KNN) [38], linear regression analysis [25], random forest (RF) [19], decision tree [26], and Support Vector Machine (SVM) [15] as the learning model. Both training and test code are implemented in Python, and the machine learning library is the open source Python machine learning framework, scikit-learn [1]. The software implementation of this method is limited by Python's low implementation efficiency and its inability to execute on embedded processors, which can only be done off-line.

The hardware implementation utilizes the PHY Analyzer module to identify the WiFi device in real time. We embed the implementation of the identification logic into the PHY Analyzer module with the help of Vivado HLS (High-Level Synthesis) tool [30]. The identification is done on the embedded software which reads data from PHY Analyzer through the AXI bus.

As shown in Figure 10, we use AD9361 [4] board with FMC interface as RF front-end device. PHY and low MAC layers are implemented on the Xilinx KC705 [5] evaluation board. Specifically, Microblaze provided by Xilinx is adopted to realize the low MAC logic, AXI bus is used to connect the hardware IPs. We use Cypress CYUSB3KIT-003 [3] explorer kit and an FMC interconnect board [2] to support the implementation of USB 3.0 communication library. Ubuntu 14.04 operating system is used on host computer. We develop verilog HDL code and C code of low MAC in Vivado 2015.2 and SDK respectively.

6.2 PHY layer information extraction

This section tests the validity of the TickSEC reading PHY layer information, we use the real 802.11 devices, selects the most commonly used 2.4 GHz and 5 GHz frequency band, and performs 1–2 minute monitoring on this frequency band to record the frequency deviation.



Figure 10 A photo of TickSEC system

Table 4 Device distribution and frequency offset of channel 1

Device No.	Last 24 bits of MAC address	Frame count	avg. Δf	σ
1	0x003A98	4247	-26.5586	6.8343
2	0xA0C589	686	5.0964	6.7770
3	0xA434D9	556	-24.5785	7.6319
4	0x66C002	507	-0.1805	4.3990
5	0x48437C	409	-11.0173	5.1095
6	0x08D40C	394	0.3264	3.7925
7	0x2ADCB6	364	-1.1527	11.1968
8	0xDE436A	229	-27.3364	3.3673
9	0x8AF314	195	-24.1565	6.2982
10	0xF48B32	151	6.2376	7.9951

Table 4 shows the devices and their frequency offset distributions collected from received frames on channel 1 (2.412GHz). It can be seen that there is a relatively large difference in frequency offset between different devices. With the exception of device 7, the standard deviation of frequency offset of the same device is small.

Table 5 shows the devices and their frequency offset distributions collected from received frames on channel 161 (5.805GHz). Compared with 2.4 GHz, there is a relatively large difference in frequency offset between different devices. With the exception of device 7, the standard deviation of frequency offset of the same device is very large, further indicating that the frequency offset between different devices is quite different.

There are two main reasons for using the frequency offset: First, the frequency offset is related to the hardware circuit of the device, and the difference between devices cannot be ignored. For other physical layer information, channel-related information such as CSI and RSSI will fluctuate after the device is moved, thus not suitable to tell between devices; and the clock offset can only be used to identify the AP and not identify the STA. Second, the frequency offset is used to show the advantages of the platform, because the frequency offset is difficult to obtain from the commercial network card.

Table 5 Device distribution and frequency offset of channel 161

Device No.	Last 24 bits of MAC address	Frame count	avg. Δf	σ
1	0xDCEF09	4953	-4.8232	9.7126
2	0x9DA868	711	-3.4210	4.7535
3	0x67A622	672	-4.0277	7.7692
4	0x9AE07C	291	-3.7384	8.9364
5	0xD6FF00	189	-3.9803	5.4527
6	0xB699D3	83	-2.8351	1.9508
7	0x7ADC74	83	-62.3400	24.0868
8	0x2C9FC7	56	-5.5038	13.2189
	Others	575	-18.3011	37.5738

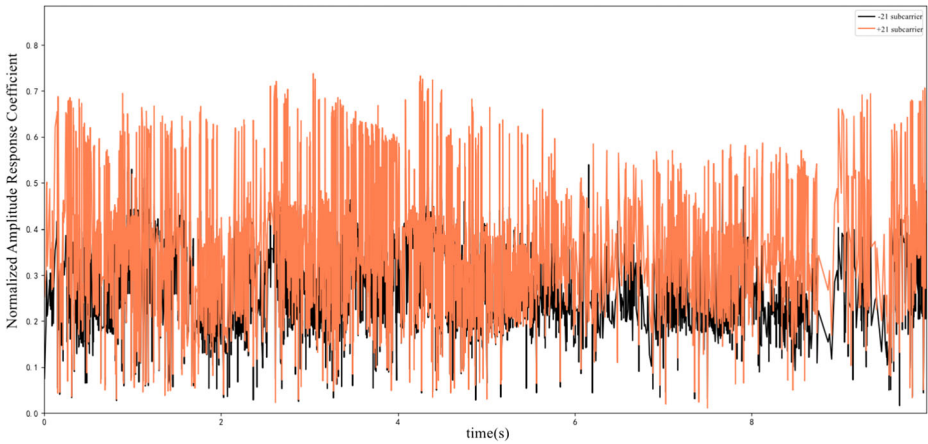


Figure 11 A device subcarrier amplitude response curve at 2.4GHz

Besides frequency offset, other data are also collected during the case study to demonstrate the effectiveness of TickSEC to read PHY layer information. CSI data is presented here as an example. Figures 11 and 12 shows the CSI of two devices at 2.4 GHz and 5 GHz over time. The two devices are the devices that receive the most frames in two channels. The number of received frames is 4247 and 4953 respectively. For a period of more than 10 seconds, the amplitude responses of the two sub-carriers -21 and $+21$ in the CSI are selected for display. These two subcarrier are also the pilot subcarrier specified in the 802.11 protocol. From Figure 11, it can be seen that the trend of the overall magnitude response is slowly changing except for the sudden drop and the sudden increase caused by the environmental change.

We also use machine learning models to analyze data off-line. Seen from Table 6 we can see that, regardless of which machine learning model we use, the correct rate is above 96%.

6.3 Result analysis of the case study

For the hardware implementation of the identification method, the scenario is as shown in Figure 13, where (A) shows whether the packets sent from all surrounding WiFi devices

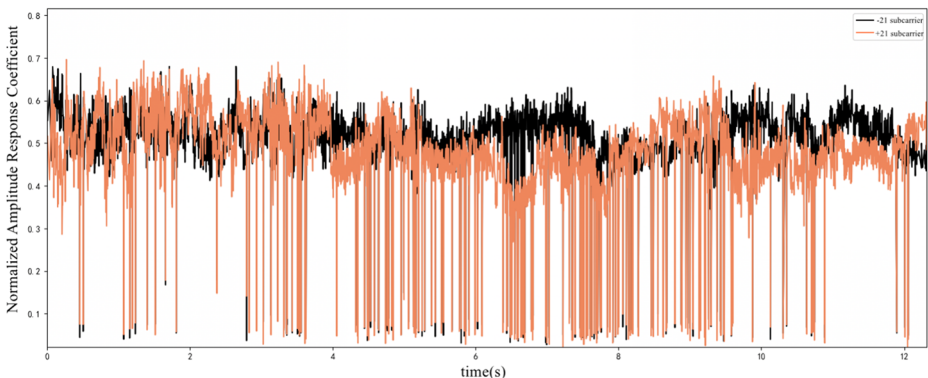
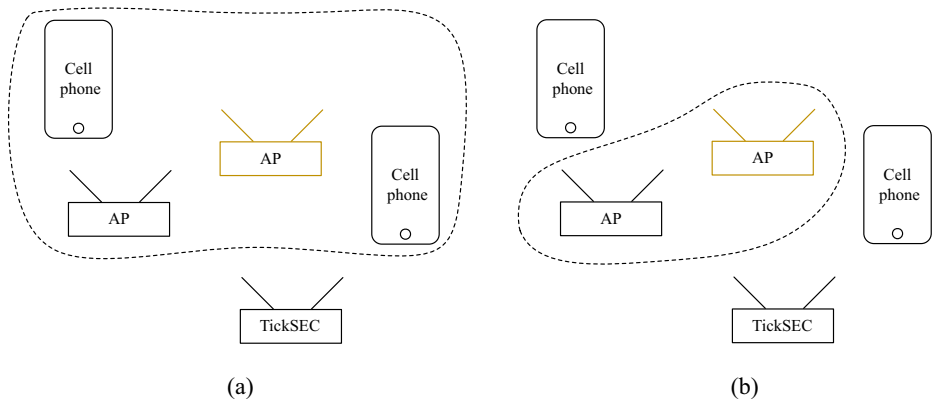


Figure 12 A device subcarrier amplitude response curve at 5GHz

Table 6 Offline correct rate of identifying different WiFi devices

Models	Correct rate (%)
KNN	98.15
Linear regression analysis	98.48
Random forest	97.31
Regression tree	96.80
SVM	97.98

**Figure 13** Real-time hardware identification of different WiFi devices**Table 7** Frequency offsets for different devices

Device No.	Frame count	avg. Δf	σ
Huawei	451	14.819	2.9401
iPhone	223	-29.0859	2.8856
Xiaomi	512	9.8436	2.3449

Table 8 Real-time identification of the correct rate of different WiFi devices

Items	Identify from all	Identify from 2 devices
Recv packets	8756	6302
Correctly identified	7984	5899
False positives	952	83
Missed reports	320	320
Correct rate	85.47%	93.61%
False alarm rate	10.87%	1.32%
Missing rate	3.65%	5.08%

are from the specified device, and (B) shows that from two “suspicious” WiFi The device determines whether it is worth the device. The WiFi channel is Channel 1 (2.412GHz), and all the devices to be identified are commercial devices under real environment.

We monitor three devices and collect 1186 trustable frames as training set. The brands of three devices are Huawei, iPhone, and Xiaomi respectively. The frequency offset statistics are as shown in Table 7. It can be seen that the frequency offset Δf between different devices is very large, and the standard deviation σ is not as great as the differences between Δf s. So it is reasonable to identifying the device with the above method.

The success rate of the recognition is shown in Table 8. It can be seen that the success rate of identifying the designated devices from all 3 devices is about 85%, and the success rate of identifying the specified devices from the two devices reaches more than 93%.

6.4 Performance evaluation

Table 9 shows the resource usage of TickSEC. Flip-Flops and look-up tables (LUTs) are the main logic resources of the Xilinx 7 series FPGAs. Flip-Flops can be used as registers. LUTs can be used for logic and storage. Called Memory LUT. BRAM (Block RAM) is the main storage resource of Xilinx 7 series FPGAs. Most FIFOs are implemented using BRAM, and a few are implemented using DRAM (Distribute RAM). DRAM actually consumes Memory LUT resources.

We can see that TickSEC has increased the use of Flip-Flop resources compared to Tick but not much. The use of other resources such as LUT and BRAM resources has actually decreased, for the following reasons:

- TickSEC uses multi-level register caching for frame alignment, so it consumes more Flip-Flop, but it is only a small fraction of the original Tick consumption, so Flip-Flop usage is not much increased;
- LUT has a small decrease because the RF and LowMAC of the original Tick system are both in the block design. TickSEC moves the PHY layer to the inside of the block design and encapsulated as an IP core, thus effectively reducing the LUT resources consumed by the connection inside and outside;
- TickSEC reduces the unnecessary depth of some FIFOs, thus optimizing the usage of BRAM and Memory LUTs;
- BUFG is used for clock division and deburring. The BUFG resource is decreased because TickSEC optimizes the global clock. the BUFGs of the same frequency clock are combine generated.

Table 9 Resource usage of TickSEC and original Tick

Resource	TickSEC		original Tick	
	Used	%	Used	%
Flip-Flop	82210	13.54	76781	12.65
LUT	97894	32.24	98657	32.50
Memory LUT	4060	3.10	4551	3.48
I/O	224	32.00	224	32.00
BRAM	264	25.63	319	30.97
DSP48	303	10.82	294	10.50
BUFG	14	43.75	17	53.12

Table 10 Latency of original Tick system

Frame size (bytes)	100	500	1000	1500	2000	3000	4000
Latency (μ s)	334	336	382	404	415	449	489

Then we test the transmission delay. If the delay is too high, the transmission performance will be greatly affected, and the time interval reserved for the user will be reduced. Table 10 contains the end-to-end processing delay of original Tick system [31]. In TickSEC, the delay only increases by about 65ns, because the well-designed hardware logic ensures that the physical layer information is prepared. The main time spent is for reading multiple sets of AXI registers, the cost is negligible.

Combined with the test results, TickSEC can extract the physical layer information effectively, and provides a variety of common physical layer information under the premise of occupying less resources.

7 Conclusion

We propose a platform for using PHY layer information to verify security authentication. The results from evaluation show that TickSEC is efficient and effective in extracting PHY layer information. It provides a variety of common PHY layer information under the premise of low resource consumption and low impact on performance. The case study demonstrates TickSEC's value in practical security authentication. In general, TickSEC can meet the needs of WiFi security researches as a verification platform. We applied the machine learning approach to the TickSEC, and we plan to add intelligence-related features in the future.

Acknowledgements The work is funded by National Key Research and Development Plan of China (2017YFB0801702) and key research project of National Natural Science Foundation (No. 61531004).

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

1. Scikit-learn. <http://scikit-learn.org> (2007)
2. Cyusb3acc-005 fmc interconnect board. <http://www.cypress.com/documentation/development-kits-boards/cyusb3acc-005-fmc-interconnect-board-ez-usb-fx3-superspeed> (2014)
3. Cyusb3kit-003 superspeed explorer kit. <http://www.cypress.com/documentation/development-kits-boards/cyusb3kit-003-ez-usb-fx3-superspeed-explorer-kit> (2014)
4. Ad9361. <http://www.analog.com/en/products/rf-microwave/integrated-transceivers-transmitters-receivers/wideband-transceivers-ic/ad9361.html> (2018)
5. Xilinx kc705 evaluation board. <http://www.xilinx.com/products/boards-and-kits/ek-k7-kc705-g.html> (2018)
6. Bagci, I.E., Roedig, U., Martinovic, I., Schulz, M., Hollick, M.: Using channel state information for tamper detection in the internet of things. In: Proceedings of the 31st Annual Computer Security Applications Conference. ACM, pp. 131–140 (2015)
7. Blossom, E.: GNU Radio: tools for exploring the radio frequency spectrum. *Linux journal* **2004**(122), 4 (2004)

8. Brik, V., Banerjee, S., Gruteser, M., Oh, S.: Wireless device identification with radiometric signatures. In: Proceedings of the 14th ACM International Conference on Mobile Computing and Networking. ACM, pp. 116–127 (2008)
9. Faria, D.B., Cheriton, D.R.: Detecting identity-based attacks in wireless networks using signalprints. In: Proceedings of the 5th ACM Workshop on Wireless Security. ACM, pp. 43–52 (2006)
10. Fortune, S.J., Gay, D.M., Kernighan, B.W., Landron, O., Valenzuela, R.A., Wright, M.H.: Wise design of indoor wireless systems: practical computation and optimization. *IEEE Comput. Sci. Eng.* **2**(1), 58–68 (1995)
11. Fuhrmann, D.R., Antonio, G.S.: Transmit beamforming for mimo radar systems using partial signal correlation. *IEEE Transaerospeletronsyst* **44**(1), 1–16 (2008)
12. Han, S.H., Lee, J.H.: An overview of peak-to-average power ratio reduction techniques for multicarrier transmission. *IEEE Wirel. Commun.* **12**(2), 56–65 (2005)
13. Huang, J., Albazraqoe, W., Xing, G.: Blueid: a Practical System for Bluetooth Device Identification. In: INFOCOM, 2014 Proceedings IEEE. IEEE, pp. 2849–2857 (2014)
14. Jana, S., Kasper, S.K.: On fast and accurate detection of unauthorized wireless access points using clock skews. *IEEE Trans. Mob. Comput.* **9**(3), 449–462 (2010)
15. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In: European conference on machine learning. Springer, pp. 137–142 (1998)
16. Kalamandeen, A., Scannell, A., de Lara, E., Sheth, A., Lamarca, A.: Ensemble: cooperative proximity-based authentication. In: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services. ACM, pp. 331–344 (2010)
17. Khattab, A., Camp, J., Hunter, C., Murphy, P., Sabharwal, A., Knightly, E.W.: WARP: a flexible platform for clean-slate wireless medium access protocol design. *ACM SIGMOBILE Mobile Computing and Communications Review* **12**(1), 56–58 (2008)
18. Li, X.: A verification platform for WiFi physical layer security research. Master's thesis, Peking University, Beijing (2017)
19. Liaw, A., Wiener, M., et al.: Classification and regression by randomforest. *R news* **2**(3), 18–22 (2002)
20. Liu, F.J., Wang, X., Tang, H.: Robust physical layer authentication using inherent properties of channel impulse response. In: Military Communications Conference, 2011-MILCOM 2011. IEEE, pp. 538–542 (2011)
21. Liu, F.J., Wang, X., Primak, S.L.: A two dimensional quantization algorithm for CIR-based physical layer authentication. In: 2013 IEEE International Conference on Communications (ICC). IEEE, pp. 4724–4728 (2013a)
22. Liu, H., Wang, Y., Yang, J., Chen, Y.: Fast and practical secret key extraction by exploiting channel response. In: INFOCOM, 2013 Proceedings IEEE. IEEE, pp. 3048–3056 (2013b)
23. Liu, H., Wang, Y., Liu, J., Yang, J., Chen, Y.: Practical user authentication leveraging channel state information (CSI). In: Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security. ACM, pp. 389–400 (2014)
24. Mao, Y., Zhang, Y., Zhong, S.: Stemming downlink leakage from training sequences in multi-user MIMO networks. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, pp. 1580–1590 (2016)
25. Olive, D.J.: Linear regression analysis. *Technometrics* **45**(4), 362–363 (2003)
26. Quinlan, J.R.: Induction of decision trees. *Mach. Learn.* **1**(1), 81–106 (1986)
27. Release, M.: The mathworks. Inc, Natick, Massachusetts, United States 488 (2013)
28. Schmogrow, R., Nebendahl, B., Winter, M., Josten, A., Hillerkuss, D., Koenig, S., Meyer, J., Dreschmann, M., Huebner, M., Koos, C.: Error vector magnitude as a performance measure for advanced modulation formats. *IEEE Photon. Technol. Lett.* **24**(1), 61–63 (2011)
29. Van de Beek, J.J., Sandell, M., Borjesson, P.O.: ML estimation of time and frequency offset in ofdm systems. *IEEE Trans. Signal Process.* **45**(7), 1800–1805 (1997)
30. Winterstein, F., Bayliss, S., Constantinides, G.A.: High-level synthesis of dynamic data structures: a case study using vivado hls. In: International Conference on Field-Programmable Technology, pp. 362–365 (2013)
31. Wu, H., Wang, T., Yuan, Z., Peng, C., Li, Z., Tan, Z., Ding, B., Li, X., Li, Y., Liu, J., et al.: The tick programmable low-latency sdr system. In: Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking. ACM, pp. 101–113 (2017)
32. Xi, W., Li, X., Qian, C., Han, J., Tang, S., Zhao, J., Zhao, K.: KEEP: Fast secret key extraction protocol for D2D communication. In: 2014 IEEE 22nd International Symposium of Quality of Service (IWQoS). IEEE, pp. 350–359 (2014)
33. Xiao, L., Greenstein, L., Mandayam, N., Trappe, W.: Fingerprints in the ether: using the physical layer for wireless authentication. In: IEEE International Conference on Communications. IEEE, pp. 4646–4651 (2007)

34. Xiao, L., Reznik, A., Trappe, W., Ye, C., Shah, Y., Greenstein, L., Mandayam, N.: PHY-authentication protocol for spoofing detection in wireless networks. In: Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE. IEEE, pp. 1–6 (2010)
35. Xiong, J., Jamieson, K.: Securearray: improving wifi security with fine-grained physical-layer information. In: Proceedings of the 19th Annual International Conference on Mobile Computing and Networking. ACM, pp. 441–452 (2013)
36. Yang, J., Chen, Y., Trappe, W., Cheng, J.: Detection and localization of multiple spoofing attackers in wireless networks. *IEEE Trans. Parallel Distrib. Syst.* **24**(1), 44–58 (2013)
37. Zeng, K., Govindan, K., Mohapatra, P.: Non-cryptographic authentication and identification in wireless networks [security and privacy in emerging wireless networks]. *IEEE Wirel. Commun.* **17**(5), 56–62 (2010)
38. Zhang, M.L., Zhou, Z.H.: MI-knn: a lazy learning approach to multi-label learning. *Pattern Recogn.* **40**(7), 2038–2048 (2007)