Real-Time Wireless Physical Layer Encryption

Brandon Z. Katz, Cem Sahin, and Kapil R. Dandekar

Drexel Wireless Systems Lab, Department of Electrical and Computer Engineering,
Drexel University, Philadelphia, PA, 19104, USA
Emails: {bzk23, cs486, krd26}@drexel.edu

***Abstract*—In today's world where inexpensive, hobbyist grade devices can be used to infiltrate 802.11 networks, wireless security must be re-examined from the ground up. Current security schemes rely on pre-shared keys or some form of centralized key server. Both use cases are vulnerable to "cheap" man-in-the-middle and brute-force attacks as well as user error. Recently, physical (PHY) layer techniques have been proposed for the generation of encryption keys on the fly without the use of vulnerable key sharing. While these PHY layer techniques have been demonstrated to work in controlled settings with offline processing, little work has been done to integrate them into existing wireless standards. In this work, we present an integration of PHY layer channel state-based encryption key generation into a real-time 802.11 compliant software-defined radio. Our implementation samples application layer traffic to determine channel state information and produces keys for use in encrypting packets. Experimental results indicate that our system successfully samples application layer traffic to generate encryption keys in real-time.**

***Index Terms*—Data security, cryptography, encryption, channel estimation, IEEE 802.11 standards**

I. Introduction

The modern exchange of information over wireless networks poses security challenges never dreamed of by early wireless pioneers. Wired Equivalent Privacy (WEP), the security standard introduced with 802.11 in 1997, was broken within four years [1]. Today, vulnerabilities in the replacement for WEP, WiFi Protected Access (WPA, WPA2), can be easily exploited using hobbyist tools such as the WiFi Pineapple, an open source 802.11 penetration tester, which costs less than $100 [2]. The WiFi Pineapple incorporates a multitude of free software in order to gain access to 802.11 networks. In addition, the device is extensible with room for new attacks against standards that have not yet been written.

One major commonality between existing encryption schemes is the use of pre-shared keys. While pre-shared keys can be strengthened through the use of longer and more random keys, they are susceptible to man-in-the-middle and eavesdropping-style attacks. In schemes such as WPA2, where the user has input to the key generation process, networks also become vulnerable to simple brute-force-style attacks due to weak passphrases. Despite our best data security efforts, we also often encounter unsecured wireless networks in public places.

One potential solution to the stated attacks, and even the public network problem, is to move security down to the physical layer of radio networks. Multiple algorithms have been proposed for the generation of symmetric encryption keys through the physical (PHY) layer using channel estimates [3], [4]. These algorithms allow two radios to form an encrypted session over an unencrypted channel while precluding eves-dropping and man-in-the-middle attacks. In addition, they can be used to defeat brute force attacks by continually changing encryption keys in a random fashion. Multiple key metrics are of utmost importance to these algorithms. The first metric is secret bit (s-bit) generation rate or the speed at which encryption bits are produced by the algorithm. This metric is an important consideration as it impacts how quickly keys can be generated at the beginning of a session and also how quickly they can be refreshed. The second major metric is s-bit error rate or how often corresponding bits on opposite ends of the channel fail to match. The error rate must be kept to a minimum as any one bit error means the entire key must be discarded, wasting time and resources.

Current algorithms used for extracting keys from wireless channels differ in their method of using the same channel state information. In [3], the peak magnitude of the channel impulse response is taken from each OFDM probe for use in a level-crossing algorithm to generate bits. This method has not only a very low s-bit error rate, but also a low s-bit generation rate (on the order of 1 secret bit per second). In addition, the tested level-crossing algorithm requires information to be sent over an unencrypted channel in order to generate the final key. In [4], OFDM channel estimates are compared by subcarrier index, thereby comparing many narrowband channels instead of one wideband channel from each probe. While this method has a very high s-bit generation rate (on the order of 1,000 secret bits per second) and does not require the transmission of algorithm data over an unencrypted link, it has a relatively high error rate. One major commonality between these demonstrated algorithms is the use of dedicated channel probes. These probing packets require specific coordination and also take up channel capacity.

As of now, these algorithms have been evaluated and shown to work only on experimental testbeds in controlled environments with offline processing [3], [4]. They have not been put to use in real-time, standards-compliant radio networks. In this paper, we present a real-time, standards-compliant system for channel-based encryption, along with initial testing results. In addition, we show that it is possible to use application (APP) layer traffic to sample wireless channels quickly enough to use for key generation as opposed to dedicated channel probes as

978-1-5090-1199-5/16/$31.00 ©2016 IEEE

have been used in the past [3], [4].

We organize our paper as follows. In Section II, we describe a technique of generating encryption keys on the PHY layer. Section III lays out our framework for using APP layer traffic to sample the wireless channel, followed by how we propose incorporating the sampling technique and key generation algorithm into a real-time radio. In Section IV, we discuss our preliminary experimental setup and results. Finally, Section V contains possible future work, followed by our conclusions in Section VI.

II. PHY Layer Key Generation Algorithm

Our integrated technique is based on the level-crossing algorithm described in [3] and is designed with the 802.11-2012 standard in mind. This base algorithm was selected as it has been thoroughly vetted and is well known. In addition, we decided that it is best to start with a focus on low s-bit error rate, as this algorithm exhibits.

Here, we summarize how the algorithm presented in [3] works. First, a probing phase is entered where the channel between two radios is sampled using probing packets. These probes are assembled at both ends of the link and once enough probes are exchanged, each node independently filters the estimated channel to reduce the presence of fast-fading. After filtering, the nodes compute the standard deviation of the channel to use as a threshold to determine if the sample at each time index is a 1, 0, or should not be considered as a bit. Next, a window is applied to the parsed bits, and a bit is considered to be present at both radios if there is a run of N same bits in a row. Finally, one radio sends the indices where it believes there is a bit to the other radio, and the other radio replies with a list of indices that it agrees contain a useful bit. Any eavesdropper would only have information about what samples are being used as bits, but not what bit the samples were parsed to. As the wireless channel is reciprocal only between the two cooperating radios, the eavesdropper would not parse the same bits, therefor leaving them with a useless key.

III. Real-Time System

A. Interrupt-Based Sampling

While current algorithms rely on dedicated probing packets [3], [4], our implementation utilizes 802.11 preamble information from APP layer traffic. The bursty and asymmetric nature of application layer traffic presents a challenge to two radios sampling the channel symmetrically, as it cannot be assumed that each received packet directly corresponds to a packet at the other participating radio. This asymmetry is overcome through the use of an internal timer interrupt on each radio such that there is an interrupt approximately once per channel coherence time interval. The coherence time is dependent on the environment and may range from tens to hundreds of milliseconds [5]. The interrupt can be viewed as a request for a packet sample. Once an interrupt occurs, the next received packet transmitted from a participating radio is used in the channel estimation process. All other packets should be disregarded for the purpose of sampling the channel, but should continue through the typical MAC pipeline. It should be noted that if the interval between received packets on any participating station is greater than the approximate channel coherence time, the channel sampling process should be put on hold as the packets used for channel estimation may not be correlated.

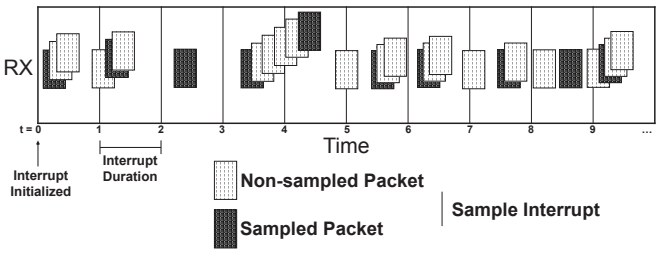

Fig. 1: Illustration of APP layer traffic sampling using interrupts

An illustration of the packet sampling mechanism is shown in Fig. 1. In this diagram, the received packets at a participating radio are shown over time. Note that the diagram is not to scale. In reality, the duration of a transmitted packet (on the order of microseconds) is much smaller than a sample window (on the order of milliseconds). To gather approximately symmetric samples, the process depicted in Fig. 1 should run on each participating receiver.

It is worth noting that in a multi-radio system, the sample request interrupt at each receiver will not occur at the exact same time. To fix this, the timer interrupt should be roughly synchronized across the network when the process begins. Sample requests will still not be at the exact same times due to imperfect synchronizing and timer interrupt variance, but they should occur within one channel coherence time window which is fast enough to generate correlated channel estimates.

B. Online Key Generation

Now that we have established a framework for sampling APP layer traffic, we can use the technique to integrate the key generation algorithm into the MAC layer of a radio with minimal overhead. First, when two radios decide to generate a key, the interrupt timer must be set on both receivers. In order to make sure the interrupts occur close together in time, a synchronization handshake is introduced. Using 802.11 as an example, a packet is sent by an access point (AP) or station (STA) to signal that it would like to generate a key. A return packet is sent either declining to enter into key generation, or accepting while also serving as a signal to start the interrupt timer.

Next, during periods with a bidirectional stream of APP layer packets, both the AP and STA collect packet samples. Once a predetermined number of packets are collected, the AP and STA suspend collection. As samples are collected, their preambles are used to generate channel state information. The samples are then processed following the algorithm

summarized in Section II between received packets to maintain MAC performance. Two more packets need to be introduced to accommodate the index transmission called for in the algorithm.

Once a key is established, the key can be continually changed by generating new bits to add to the end of the existing key. While not exactly the same, this key modification operation may be reminiscent of the key changing described in the 802.11i-2004 amendment as part of the Temporal Key Integrity Protocol (TKIP) [6]. Key mismatches can be determined through the sending of known packets, such as management frames. If the keys at each radio turn out to be mismatched, the process can be started over or the key can be reverted. If bidirectional traffic is not maintained, the radio with a sample underflow should signal the other to suspend key generation.

Algorithm 1 shows a pseudocode implementation of our online key generation process. The pseudocode does not include signaling packets, but assumes that the process begins as soon as the radio is turned on.

Algorithm 1 Real-Time Sampling Algorithm

```
1: global pktSample          ▷ / keep track of if a sample is needed and how many are needed
2: global numSamples         ▷ / keep track of how many samples have been collected
3: static maxNumSamples ▷ / number of samples needed to generate a key
4:
5: procedure ONSAMPLEINTERRUPT
6:     pktSample++           ▷ / increment number of samples needed
7: end procedure
8:
9: procedure MACHIGHPKTRX(packet)
10:    if packetIsGood && pktSample > 0 then
11:        if pktSample > 2 then
12:            suspend sample collection
13:        else if numSamples > maxNumSamples then
14:            parse key from stored samples using level crossing algorithm
15:        else
16:            store channel sample from packet
17:            pktSample– ▷ update need for a new sample
18:        end if
19:    end if
20:    proceed with standard mac packet processing
21: end procedure
22:
23: procedure MAIN
                ▷ / Initialize transmit and receive along with standard MAC processing
24:    init sampleInterrupt      ▷ / Initialize the interrupt timer
25:    while 1 do
26:                              ▷ / Wait for packet TX or RX
27:    end while
28: end procedure
```

IV. Experimental Evaluation

A. Setup

The procedures described in Section III were integrated into the Wireless Open-Access Research Platform (WARP) 802.11 reference implementation on WARPv3 boards [7] for verification and testing. Primary testing was carried out using two nodes, an AP and an STA. All processing was performed on the WARP hardware, either on the FPGA directly or on a MicroBlaze softcore. The two nodes were set up approximately ten feet apart in a research lab with mostly office space. Experimentation was carried out over the air on both congested as well as unoccupied channels. Standard 2.4 GHz monopole antennas were used so as to resemble a typical consumer wireless network. APP layer traffic was generated using both ping commands and file transfers between host PCs attached to the AP and STA. For the purpose of this test, the sampling period used was 80 ms. In order to facilitate sampling, APP layer data was sent at a rate such that packets were sent much faster than once per coherence time, nominally 12 Mbps.

Keys were generated using a window over the channel state information derived from the preamble of the 802.11 packets. Groups of 80 packets were considered at a time. This grouping allowed full keys to be generated even if a bit error occurred during one segment of the process. If one segment of the key was discovered to be mismatched through the reception of a non-decodable test packet, the key was reverted by a segment and the process continued. In addition to practicality, this operation mirrors the possible use case of changing a key over time. Success was measured by bit error rate as well as raw bit generation speed.

Standards compliance was spot-checked though the use of a commercial WiFi device. A mobile phone was attached to the modified AP and the internet was accessed in various configurations. Spot-checking occurred while key generation was in progress with the modified STA as well as while the modified STA was not connected to the modified AP.

B. Results and Discussion

Initial testing results are summarized in Table I below.

Table I: Summary of experimental results using modified WARP system

Experiment duration	20 min
Interrupt timer	80 msec
Average s-bit rate	0.63 s-bits/sec
Average bit-error rate	6.7%

Overall, the channel-based key generator added into the WARP 802.11 reference implementation generated symmetric bits with a secret bit rate of 0.63 s-bits/sec and with an average bit error rate of 6.7%. While this error rate seems low, a single

bit mismatch renders the entire key segment useless. As a result, the overall segment mismatch rate was close to 33.3% with the average segment containing five bits generated from 80 packet samples. This statistic means that for a complete key of 128 bits, 35 key segments had to be generated on average with 10 of the segments discarded due to the presence of a bit error. While the bit generation rate of this system is lower than research implementations [8], [9], we believe that this is acceptable as this algorithm can continually run, without introducing the overhead needed for dedicated probing packets, so long as sufficient APP layer traffic is moving across the network.

Some processing overhead was introduced to the WARP 802.11 MAC as a result of running the algorithm. Round-trip latency for APP layer traffic was increased from an average of 0.85 ms to 1.21 ms, representing a 42.4% increase. This increase can be largely attributed to the processing necessary to extract keys from the channel state information. The algorithm was run on the same MicroBlaze processing core used for MAC High functions on the WARP 802.11 reference as opposed to an isolated processing environment. No change in effective data throughput was observed after the introduction of the key extraction algorithm to the WARP 802.11 reference.

C. Augmentation of Existing Encryption Techniques

In the current implementation, this technique seems to be best suited for augmenting WPA2 or similar processes. A sample use case could involve starting with a passphrase and then using this algorithm to change the key over time in a random fashion. This process can be equated to using the WPA2 passphrase as a means of authentication and initial encryption, and then shifting the encryption responsibilities over to the physical layer security algorithm. This technique also has potential to augment standards like HTTPS in unsecured networks. Even in places where the 802.11 network is unsecured, this method can be used to secure individual user connections and can generate enough bits to make keys from scratch. A slower key generation rate could be traded for lower bit error rate to ensure fully symmetric keys are generated on the first try.

V. Future Work

While this system has shown that it is possible to generate PHY layer encryption keys in a standards-compliant environment using application layer traffic, more work has to be performed to determine the optimal sampling speed and algorithm confidence parameters. Future work may include an automatic gain control style system to determine the optimal packet sample rate for the given channel. This feature may include the incorporation of real-time channel coherence time estimation similar to what is discussed in [10]. Future system testing will need to be conducted in a variety of measured and emulated environments to quantify key error rates and randomness in different situations.

VI. Conclusion

In this work, we presented our technique for interrupt-based sampling and a framework for integrating a channel-based key generation scheme into an 802.11 reference design. Through experimentation, we demonstrated that it is possible to use regular APP layer traffic as a means of sampling wireless channels quickly enough to generate symmetric keys based on the channel state information. Verification was conducted using WARP software-defined radios over real air channels. Overall, we were able to establish key generation rates and key error rates that are promising for future research and commercial expansion.

VII. Acknowledgments

This material is based upon work supported by the National Science Foundation under grant numbers CNS-1228847 and CNS-1422964.

References

[1] S. R. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4," in *Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography*, ser. SAC '01. London, UK, UK: Springer-Verlag, 2001, pp. 1–24. [Online]. Available: http://dl.acm.org/citation.cfm?id=646557.694759

[2] "WiFi Pineapple." [Online]. Available: https://www.wifipineapple.com

[3] S. Mathur, W. Trappe, N. Mandayam, C. Ye, and A. Reznik, "Radio-telepathy: Extracting a Secret Key from an Unauthenticated Wireless Channel," in *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, ser. MobiCom '08. New York, NY, USA: ACM, 2008, pp. 128–139. [Online]. Available: http://doi.acm.org/10.1145/1409944.1409960

[4] C. Sahin, B. Katz, and K. R. Dandekar, "Secure and Robust Symmetric Key Generation using Physical Layer Techniques under Various Wireless Environments," in *Radio and Wireless Symposium (RWS), 2016 IEEE*, Jan 2016.

[5] H. MacLeod, C. Loadman, and Z. Chen, "Experimental studies of the 2.4-GHz ISM wireless indoor channel," in *Communication Networks and Services Research Conference, 2005. Proceedings of the 3rd Annual*, May 2005, pp. 63–68.

[6] "IEEE Standard for information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 6: Medium Access Control (MAC) Security Enhancements," *IEEE Std 802.11i-2004*, pp. 1–190, July 2004.

[7] "WARP project." [Online]. Available: http://warpproject.org

[8] S. Jana, S. N. Premnath, M. Clark, S. K. Kasera, N. Patwari, and S. V. Krishnamurthy, "On the Effectiveness of Secret Key Extraction from Wireless Signal Strength in Real Environments," in *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '09. New York, NY, USA: ACM, 2009, pp. 321–332. [Online]. Available: http://doi.acm.org/10.1145/1614320.1614356

[9] M. Wilhelm, I. Martinovic, and J. Schmitt, "On key agreement in wireless sensor networks based on radio transmission properties," in *Secure Network Protocols, 2009. NPSec 2009. 5th IEEE Workshop on*, Oct 2009, pp. 37–42.

[10] T. Yucek, R. Tannious, and H. Arslan, "Doppler spread estimation for wireless OFDM systems," in *Advances in Wired and Wireless Communication, 2005 IEEE/Sarnoff Symposium on*, April 2005, pp. 233–236.