

# A PERFORMANCE ANALYSIS OF THE SPRiNG PROTOCOL THROUGH SIMULATION

Kevin Richardson and John A. Hamilton, Jr.  
Department of Computer Science and Software Engineering  
Auburn University  
Email: {richakp,hamilton}@eng.auburn.edu

Martin C. Carlisle  
Department of Computer Science  
United States Air Force Academy  
carlisle@acm.org

Keywords: WEP, SPRiNG, OPNET, Network Simulation.

## ABSTRACT

Many of today's organizations have set up wireless networks for company use, some merely for the convenience of not having wires everywhere. In addition, many of these networks are using a hardware implementation of the IEEE 802.11 WEP protocol for security. With the known vulnerabilities in WEP, these organizations are faced with either a costly upgrade or just ignoring the vulnerabilities. However, the SPRiNG protocol is a proposed solution to securing wireless communication designed to work with WEP hardware devices. Using simulation, this paper provides a performance analysis of the SPRiNG protocol's increased overhead. We determine that SPRiNG will perform acceptably with up to 20% packet loss.

## 1.0 INTRODUCTION

The advantages of having a wireless network are abundantly apparent to an average user. However, the security risks associated with wireless networks are,

unfortunately, either unknown or not considered by this set of users. This is mainly a problem with smaller organizations, such as privately owned businesses, because these companies do not typically hire network administrators. Many of these companies, unaware of the risk of intrusion, may erroneously believe they are not at risk because they do not send out any sensitive information when using the Internet.

The Wired Equivalent Privacy protocol (WEP) has the security goals of access control, confidentiality, and data integrity. Unfortunately, it does not meet any of these goals. One proposed solution to securing wireless communications is the Synchronized Pseudo Random Number Generation protocol (SPRiNG)[5]. The SPRiNG protocol solves the problems inherent in WEP but does involve some extra overhead. The added overhead comes from the need to keep the random number stream between a host and the access point synchronized. Pepyne, Ho, and Zheng, when they proposed SPRiNG, did not provide any analysis on how SPRiNG will affect performance. The purpose of this paper is to determine the extent of the

performance loss due to the increased overhead imposed by the SPRiNG protocol.

Section 2 of this paper gives a description of the problems with WEP. Section 3 follows with an explanation of the SPRiNG protocol. The experimental design is presented in Section 4 followed by a validation of the models in Section 5. The results of the experiments are given in Section 6, and conclusions in Section 7.

## 2.0 PROBLEMS WITH WEP

The drawbacks of WEP have been identified in numerous papers [1], [2], [3], [7]. The majority of the vulnerabilities in WEP are a result of the misuse of the RC4 stream cipher. The main problem is key stream reuse, which allows hackers to use cryptanalytic strategies to find out the secret key used for encryption. Knowing this key provides an adversary with the means to access the network.

Borisov, Goldbert and Wagner point out that there are actually many inherent causes of key stream reuse in WEP[2]. The most prevalent issue is that the WEP standard does not specify how the initialization vectors (IV's) used in RC4 should be selected. In fact, most devices set their IV value to 0 at the beginning of each session. This guarantees key stream reuse. The situation is made worse by the size of the IV value in each frame. Using only 24 bits guarantees key stream reuse at least every  $2^{24}$  frames, if the session continues that long. This is not unheard of when dealing with large FTP transfers. However, monitoring traffic from multiple sessions with an access point results in the demise of WEP security much faster since most sessions begin with an IV value of 0 and count upward as the session continues.

Borisov, Goldbert and Wagner also show why key stream reuse is so detrimental to the security of the wireless system[2]. The reception of two frames that were encrypted using the same IV is all that is needed to decipher the plaintext, which results in revealing the secret key. These keys have to be manually set on each mobile device in the network so they are not changed very often. The result is that an adversary has access to the network for an indefinite amount of time.

Access control is obviously broken in WEP due to these findings, but it shows the lack of confidentiality and data integrity as well. Confidentiality is compromised because knowledge of the key allows an attacker to decrypt and read any intercepted message. Data integrity is forsaken in the same manner because an attacker can intercept the message, decrypt it, modify it, and then forward it on to the recipient.

It should also be noted that Borisov, Goldbert, and Wagner describe how message authentication can be circumvented through the CRC-32 checksum used in WEP[2]. This problem allows modifications to be made to frames without detection. Furthermore, only partial knowledge of the contents of the frame is needed. The worst part is that these modifications can be made where the frame will still pass the CRC checksum.

## 3.0 SPRiNG PROTOCOL

The SPRiNG protocol [5] has the same security goals in its design as WEP. The difference is in how these security goals are accomplished. WEP relies on encryption to achieve security. In contrast, SPRiNG relies on authentication. SPRiNG uses the same RC4 stream cipher for encryption but it is

used as a secondary security measure. Therefore, SPRiNG can be implemented with a software or firmware upgrade, depending on the device. This makes SPRiNG attractive, as it does not require an expensive replacement of the hardware infrastructure.

SPRiNG stands for synchronized pseudo random number generation, which describes the basis of the protocol's functionality. In other words, every machine on the network is equipped with the same pseudo random number generator or PRNG that is used for the stream of authentication keys. Each machine on the network still needs to have the same initial seed for number generation but a key that is set in software is much easier to distribute. The authors suggest that the initial seed be generated and agreed upon at the start of each session to avoid key stream reuse.

The protocol works as follows. Assume that device A wants to communicate with device B. Device A stores the current number in the synchronized PRNG stream in the frame to be sent. The frame is then encrypted and sent to device B. When device B receives the frame, it is decrypted and a check is performed on the authentication key. If the authentication key matches device B's next number in its PRNG sequence, the frame is accepted. Otherwise, the frame is rejected. This strategy provides data authentication because it is hard to predict the outputs of a PRNG as well as key stream reuse protection because a PRNG has such a long stream of non-repeating values.

The PRNG used is a minimal standard linear congruent PRNG. Park and Miller[4] provide a description of this type of PRNG. The function used in this case is of the form:

$$f(z) = (a * z) \bmod m$$

where mod is the modulus operation. Values for  $a$  and  $m$  must be chosen carefully to ensure that every number in the range  $[1, m - 1]$  inclusive is generated before a duplicate value is seen. A PRNG that has this quality is referred to as a full period PRNG.

It must be assumed that attackers know the values of  $a$  and  $m$  in the equation used for the PRNG. This means that the attacker only needs the current seed to be able to access the network. The system is compromised from that point on if an attacker has this information because he/she can then generate all of the subsequent keys. To avoid this situation, the frame counter  $k$  is concatenated on to the PRNG output and this value is stored where WEP would store the IV value. This means it will be used as part of the encryption key. Recall that the IV values can be obtained, giving away the encryption key. This is a problem for WEP because the same encryption key is used for every frame. However, SPRiNG encryption keys will be unique, thus defeating the cryptanalytic strategies used to break WEP.

Lost frames are a potential problem with the protocol because loss can cause devices A and B to lose synchronization. To combat this problem a look-ahead window of size  $N$  is implemented. If device B receives a frame with an authentication key that does not match the next value from its PRNG, device B will look up to  $N$  numbers ahead in its PRNG sequence to decide whether or not to keep the frame. If the frame is rejected, device B does not update its seed in the PRNG sequence. If the frame is accepted, device B sets its seed to the authentication key used in the packet. The effect of this is

to ensure synchronization between the PRNG outputs of devices A and B.

It should be noted that the attacks described above that involve the use of the CRC-32 checksum in WEP are not addressed in SPRiNG. This means that an attacker that has somehow found out the current seed in the PRNG sequence can intercept and manipulate frames, thus forsaking integrity and/or confidentiality. This is much more of a problem for WEP though because there are numerous scripts on the Internet that can break WEP in a matter of minutes. The reasoning behind leaving this vulnerability in SPRiNG is that the authors of SPRiNG wanted their protocol to work on existing 802.11 hardware. However, the authors do point out that the use of an integrity checker called MIC developed for use in the Temporal Key Integrity Protocol (TKIP) could provide more security.

The improvements over WEP are apparent in the SPRiNG protocol. This is because encryption keys in WEP are generated from the concatenation of the WEP secret key and the IV value, with the IV value being passed in the clear. Thus, any two frames with the same IV are guaranteed to have been encrypted with the same key. On the other hand, encryption keys in SPRiNG are generated from the concatenation of the PRNG outputs and the frame counter, with the frame counter being passed in the clear in place of WEP's IV value. Thus, two frames with the same frame counter value have not necessarily been encrypted with the same key, especially if initial seeds are generated and agreed upon at the beginning of each session as the SPRiNG authors suggest.

The question then involves the performance of the protocol. If devices A

and B lose synchronization, a re-synchronization process must be used to continue communication. There is no communication of data while devices are synchronizing themselves so a large number of re-synchronizations can be very costly in terms of performance. This simulation compares six scenarios in terms of average end-to-end delay, the average number of re-synchronizations per second, and the average throughput in accepted data frames per second. The six scenarios include a SPRiNG implementation with normal network loss and similar networks with added 10%, 20%, 30%, 40%, and 50% frame loss probabilities.

## **4.0 EXPERIMENTAL DESIGN**

This simulation was conducted using the OPNET simulator. The designs of the OPNET node models for the source and access point are given in Section 4.0.1. The OPNET network model as well as the designs of the six scenarios implemented follows in Section 4.0.2.

### **4.0.1 Model Design**

#### **4.0.1.1 Source Model**

The source model consists of a frame generator, a processor, a wireless transmitter, a wireless receiver, and an antenna. The frame generator sends generated frames to the processor with a Poisson distribution used to determine the frame inter-arrival times. The processor then sends the frame out through the wireless transmitter. A stop and wait protocol is used in the source model so the processor does not accept any generated frames from the frame generator until it receives an acknowledgement, a negative

acknowledgement, or times out. The timeout value was set to 3 seconds.

Each regular frame sent by the source model contains the next authenticator variable in the random number stream. If the source model times out, the next frame sent will contain the next authenticator variable in hopes that the look-ahead functionality will provide the authentication. If the source model receives an acknowledgement, the next frame is sent in the same manner. If the source model receives a negative acknowledgement, a re-synchronization process begins. The SPRiNG authors do not recommend a particular re-synchronization strategy. Any re-synchronization strategy would at least require a synchronization frame being sent from the source and a synchronization acknowledgement being sent back from the access point. This is the process used in this simulation. Since this simulation is not examining the security of the SPRiNG protocol, avoiding key stream reuse with regard to re-synchronizations was not taken into account. Once the access point receives a synchronization frame, it resets its PRNG seed value to the initial seed. The source begins again with the initial seed as well once it receives a synchronization acknowledgement. This process keeps the PRNG sequence synchronized between the two devices yet ensures key stream reuse. However, the functionality of the SPRiNG protocol is maintained.

#### **4.0.1.2 Access Point Model**

The access point model consists of a processor, a wireless transmitter, a wireless receiver, and an antenna. When the wireless receiver gets a frame, it is sent to the processor. The processor then checks the frame counter and the authenticator variable

against what it is expecting, using the look-ahead functionality if needed, with the value of  $N$  set to 8. If the frame is accepted, an acknowledgement is sent to the source of the frame. If the frame is rejected, a negative acknowledgement is sent to the source of the frame and the access point prepares for the re-synchronization process.

#### **4.0.2 Scenario Design**

The network used in each scenario consists of an access point and four sources. Each source is set 25 feet away from the access point, one to the north, one to the south, one to the east, and one to the west. Each run simulates 12 hours of network activity. The first scenario is representative of normal wireless network traffic loss. The remaining scenarios that introduce an added 10%, 20%, 30%, 40%, and 50% probability of network loss respectively use a sampling from a Poisson distribution to determine whether a frame should be lost or not. A Poisson distribution was chosen to provide bursts of loss.

The statistics gathered during each simulation include the average end-to-end delay for the four sources, the average throughput at the access point in terms of the number of accepted data frames per second, and the average number of re-synchronizations per second at the access point. End-to-end delay is only calculated when an acknowledgement is received; therefore, any timeout or re-synchronization will add to the end-to-end delay time.

### **5.0 VALIDATION**

The validation of the PRNG used consists of ensuring that a full period is generated by the implemented function. To achieve this validation, a simple Java

application is used. The code tests the PRNG by creating an array that will store  $m - 2$  values (because 0 and  $m$  are excluded from the generated sequence) and filling it with the outputs of the PRNG. The array is then sorted and a check is made to see if any two consecutive numbers in the sorted array are the same. If not, the PRNG is a full period PRNG. Otherwise, it is not. We proved the PRNG used in this simulation is a full period PRNG using this application.

The next validation technique used is called internal validity and is outlined by Sargent[6]. This technique involves varying internal parameters, such as the simulation seed, to determine if they have an effect on the simulation results. To accomplish this, each scenario in the simulation was run with multiple seed values. The results from each scenario with different seeds were compared to identify any discrepancies. Since all of the results varied only slightly, this simulation was determined to be valid with respect to internal validity.

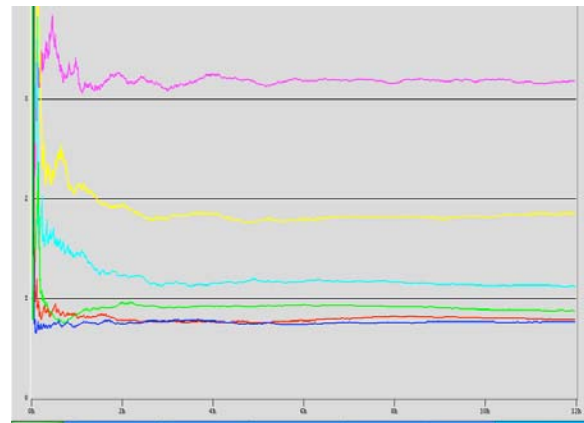
The final validation technique used was a complete packet trace, also described by Sargent[6]. The source and access point models were made to print detailed information about each frame encountered to text files to facilitate this validation technique. The source models print the authentication variable sent, what the authentication variable is being updated to, and whether the result was a timeout, acknowledgement, or negative acknowledgement for each frame sent. The access point prints four files, one for each source. The files contain the expected authenticator variable for that source, the received authenticator variable from that source, and whether the frame was accepted or rejected. These files present a way to fully trace through every event in the

simulation. Inspection of these files showed that this is a valid simulation of the functionality of the SPRiNG protocol.

## 6.0 RESULTS

The x-axis in the graphs presented in Figure 1, Figure 2, and Figure 3 all correspond to simulation time. The simulation lasts for 12 hours so each hash mark represents the passing of 2 hours. Each of these graphs also uses the following legend:

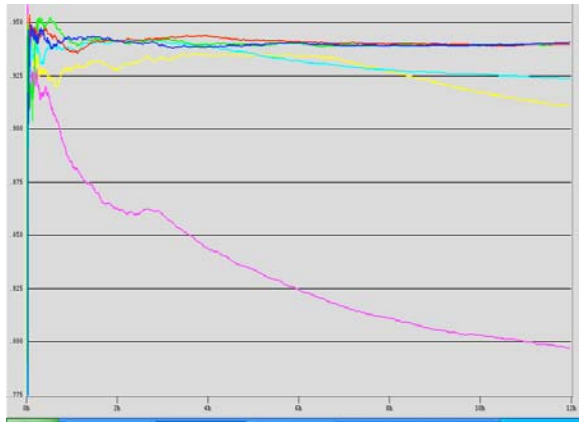
	Normal Loss
	10% Added Loss
	20% Added Loss
	30% Added Loss
	40% Added Loss
	50% Added Loss



**Figure 1 – Average End-to-End Delay for All Six Scenarios**

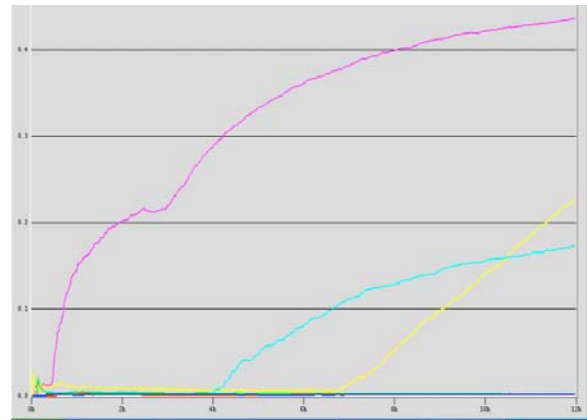
Figure 1 shows the average end-to-end delay in seconds for each of the six scenarios. This graph shows that there was very little performance difference between the normal loss, 10% added loss, and 20% added loss scenarios. A slight performance hit was experienced with the 30% added loss scenario. The 40% added loss scenario showed an average end-to-end delay value

of around 1.8 seconds. Since the timeout value was set to 3 seconds, this means that some actual communication was occurring. However, the 50% added loss scenario showed an average end-to-end delay value of around 3.2 seconds. This means that the system was constantly attempting to re-synchronize.



**Figure 2 – Average Throughput in Terms of Accepted Frames per Second for All Six Scenarios**

Figure 2 shows the average throughput in terms of accepted frames per second for each of the six scenarios. Again, the added 10% loss and added 20% loss scenarios showed very little difference from the normal loss scenario. This graph also shows that communication was able to occur with the 30% added loss and 40% added loss scenarios. It is also no surprise that the 50% added loss scenario shows a lack of communication.



**Figure 3 – Average Number of Re-synchronizations per Second for All Six Scenarios**

Figure 3 shows the average number of re-synchronizations per second for each of the six scenarios. This graph shows that the normal loss, 10% added loss, and 20% added loss scenarios had to deal with only a small number of re-synchronizations. The 30% added loss and 40% added loss scenarios did spend some time re-synchronizing, but, once again, some communication was accomplished. The increase in the number of re-synchronizations per second for the 30% added loss scenario at the four-hour mark is due to unusual bursts of loss that sent one of the sources into a long re-synchronization phase. The 40% added loss scenario made it until around the seven-hour mark before it experienced a large enough burst of loss to send three of the four sources into similar re-

synchronization phases. As expected, this graph shows that the 50% added loss scenario spent most of its time trying to re-synchronize.

## 7.0 CONCLUSIONS

The results of the simulation experiments performed for this paper show that the SPRiNG protocol can be implemented, without requiring a hardware upgrade, on a WLAN with little to no noticeable performance loss. This is due to the fact that little performance loss was experienced with regard to average end-to-end delay, average throughput in terms of accepted frames per second, and the average number of re-synchronizations per second even with 20% more than normal network loss. However, it should be noted that the performance and security of this system could be affected by the re-synchronization process used. A usable re-synchronization process must be fast, but, more importantly, must avoid key stream re-use.

## REFERENCES

[1] Boland, Hamid; Mousavi, Hamed. "Security Issues of the IEEE 802.11b Wireless LAN" Canadian Conference on Electrical and Computer Engineering, Vol. 1. May, 2004. pp. 333-336.

[2] Borisov, Nikita; Goldbert, Ian; Wagner, David. "Intercepting Mobile Communications: The Insecurity of 802.11"

Proceedings of the 7<sup>th</sup> Annual International Conference on Mobile Computing and Networking. Rome, Italy. 2001. pp. 180-189.

[3] Cam-Winget, Nancy; Housley, Russ; Wagner, David; Walker, Jesse. "Wireless Networking Security: Security Flaws in the 802.11 Data Link Protocols" Communications of the ACM, Vol. 46, Issue 5. May, 2003. pp. 35-39.

[4] Park, Stephen K.; Miller, Keith W. "Random Number Generators: Good Ones Are Hard To Find" Communications of the ACM, Vol. 31, Issue 10. October, 1988. pp. 1192-1201.

[5] Pepyne, David L.; Ho, Yu-Chi; Zheng, Qinghua. "SPRiNG: Synchronized Random Numbers for Wireless Security" IEEE Wireless Communications and Networking, Vol. 3. March 2003. pp. 2027-2032.

[6] Sargent, Robert G. "Verification and Validation of Simulation Models" Proceedings of the 30<sup>th</sup> Winter Simulation Conference. Washington D.C. 1998. pp. 121-130.

[7] Stubblefield, Adam; Ioannidis, John; Rubin, Aviel D. "A Key Recovery Attack on the 802.11b Wired Equivalent Privacy Protocol (WEP)" ACM Transactions on Information and System Security, Vol. 7, Issue 2. May 2004. pp. 319-332.