sustainability

MDPI

Article

Sustainable and Practical Firmware Upgrade for Wireless Access Point Using Password-Based Authentication

Jaejin Jang and Im Y. Jung *

School of Electronics Engineering, Kyungpook National University, Daegu 41566, Korea; jaejin@knu.ac.kr
* Correspondence: iyjung@ee.knu.ac.kr; Tel.: +82-53-950-7237

Academic Editors: James Park and Han-Chieh Chao
Received: 22 June 2016; Accepted: 23 August 2016; Published: 31 August 2016

Abstract: Wireless access points (WAPs) are devices that provide Internet connectivity to devices such as desktops, laptops, smartphones, and tablets. Hence, it is important to provide sufficient availability to devices and security for the traffic that is routed by a WAP. However, attackers can decrease the network bandwidth or obtain the traffic including private data such as search histories, login information, and device usage patterns by exploiting the vulnerabilities in firmware upgrades to install malicious firmware. To address this problem, we propose a sustainable and practical firmware upgrade for a WAP using password-based authentication. The proposed upgrade protocol ensures security by adding freshness to the firmware whenever a firmware upgrade occurs. This freshness is different for each event and each firmware; therefore, even if the freshness of one firmware is exposed, the others are secure. In addition, confidentiality, integrity, and authentication are ensured. Furthermore, the proposed protocol can be easily implemented and adapted to WAPs. Experiments are performed to evaluate the upgrade time, resource usage, and code size in wired and wireless connected environments by implementing a prototype and analyzing the security of the protocol. The results show that the proposed upgrade is secure and practical.

Keywords: practical firmware upgrade; password-based authentication; sustainable wireless access points

1. Introduction

Internet usage is currently widespread and common, and it enables us to perform many activities such as searching for information, buying products, and participating on social networks. To support the conveniences enabled by the Internet, most information and communication technology (ICT) products provide network functions such as Wi-Fi and Ethernet.

A wireless access point (WAP) is a device that enhances network availability by supporting NAT [1]. Desktops, laptops, smartphones, and tablets can communicate on internal or external network devices when connected to a WAP. ICT products have become diversified and common; thus, WAPs have also become common devices for providing network functions. WAPs can often be found in public places.

Many devices now need to connect to the Internet [2]. Hence, it is important that WAPs provide sufficient network availability, i.e., bandwidth. Traffic that is routed by WAPs also needs to be protected because it includes private information such as search histories, login information, and device usage patterns. Therefore, various security functions such as ping disable, ARP spoofing protection, and SYN flooding protection are now used in WAPs and updated via firmware upgrades. A firmware upgrade can maintain WAP functionality by providing better functions than those of the previous firmware.

However, if the purpose of the firmware upgrade is to enhance WAP functions, the firmware upgrade itself is vulnerable. An attacker can decrease network bandwidth or disable security functions by exploiting the vulnerabilities in the firmware upgrade to install malicious firmware [3,4]. This is not simply a problem for WAPs. USB devices, network interface cards, and batteries are also vulnerable to the same problem [5–7]. In addition, firmware modification attacks will become more popular because antivirus software and operating systems cannot reveal firmware-level exploits [8].

WAPs are especially vulnerable devices. WAPs can send traffic using their own network functions. In addition, they can collect information from various and multiple locations because they are installed in a large number of places. Recently, they have also been used to perform DDoS attacks [9]; thus, secure firmware upgrades are required to prevent these attacks.

In this paper, we propose a sustainable and practical firmware upgrade protocol for a WAP using password-based authentication. The proposed upgrade has two major advantages: practicality and security. We designed the upgrade by utilizing established protocols; hence, it does not need the development of new software or hardware. In addition, the upgrade process is secure because it ensures confidentiality, authentication, integrity, and freshness.

The proposed protocol provides secure firmware upgrades using Transport Layer Security (TLS) and the Salted Challenge Response Authentication Mechanism (SCRAM). Hence, it can be easily implemented and adapted for WAPs. In order to agree on the key for freshness, it uses the anonymous Diffie–Hellman (DH) key agreement protocol. However, this protocol has a typical weakness: man-in-the-middle (MITM) attacks, but the proposed upgrade can prevent MITM attacks by verifying the firmware server certificate using TLS [10,11]. In addition, we consider the sustainable and practical requirements of all processes from the initial firmware installation to the completion of the firmware upgrade. We performed experiments to evaluate the upgrade time, resource usage, and code size in both wired and wireless connected environments by implementing a prototype using Raspberry Pi B+ and Raspberry Pi. In addition, we analyzed the security of the protocol. The experimental results and analyses show that the proposed upgrade protocol is adaptable to a WAP.

The remainder of this paper is organized as follows. Section 2 reviews several previous works on firmware upgrades and compares them with the proposed upgrade process. In Section 3, we describe the firmware upgrade network architecture, practical assumptions, and protocol. In Section 4, we present the results of the performance experiments and analyze the security properties. Section 5 concludes this paper and discusses the plans for future work.

2. Related Works

There are several related studies on firmware upgrades. We explain the main ideas of these approaches and describe their advantages and disadvantages.

Odat et al. [12] studied automotive firmware upgrades and focused on reducing the upgrade costs. In their system, the firmware is upgraded using Wi-Fi of a car that already has the latest firmware. Hence, car owners do not need to return to a dealership for a firmware upgrade, which is approximately USD 150 on average, and the cost of an upgrade using the proposed system depends on the number of users. More users result in lower firmware upgrade costs. However, they only proposed a simple TCP three-way handshake that is the same as the original TCP three-way handshake for a secure connection between cars [13]. No security function for the secure connection was proposed. Furthermore, the problems of unauthorized access and external attacks were not addressed.

Tsague et al. [14] proposed a secure firmware upgrade protocol for point of sale (POS) terminals. They assumed that the portal base station has a pair of public and private keys and that the POS terminals preload the public key of the portal. The firmware is fragmented, and each fragment forms a hash chain to ensure integrity. The fragmented data is further encrypted with a pre-shared symmetric key for confidentiality. Furthermore, the portal signs the first fragment of the firmware with a private key. POS terminals can then verify whether the firmware originates from the correct portal using the preloaded portal's public key. For mutual authentication, they used Diffie–Hellman-based

authentication. However, the Diffie–Hellman protocol is commonly vulnerable to MITM attacks, and there is no method to protect against them [15]. In addition, they did not consider the exponential computation necessary for this system, which requires significantly more resources than XOR operations. Furthermore, there was no method for exchanging the symmetric encryption key.

Choi et al. [8] studied firmware validation and upgrades for home networks. In their paper, the network architecture for the firmware upgrade consists of a firmware server; a home network manager, which is a router or WAP; and home network devices. The firmware is transmitted from the firmware server to the manager and repetitively transmitted from the manager to the devices that need firmware upgrades. Of these, the transmission from the firmware server to the manager operates in a network architecture that is similar to our firmware upgrade network architecture. The firmware server and manager agree on a session key for symmetric encryption to provide confidentiality. Each firmware server and manger has a pair of public and private keys. However, Choi et al. assume that the manager key is calculated by a firmware server and transmitted to the manager. In fact, this is a very dangerous assumption because this transmission occurs before the firmware server and manager agree on the session key. Hence, if an attacker eavesdrops on the channel, he/she can obtain the private key of the manager. An attacker can then decrypt the data on the channel. Hence, this system does not ensure the confidentiality of the firmware upgrade protocol.

Steger et al. [16] studied automotive software upgrade systems. They focused on an effective upgrade rather than security using IEEE 802.11s. They evaluated its performance under various conditions because wireless communication is strongly affected by the environment. The security of the system will be studied in future work, and they plan to use SHIELD to analyze the security of the protocol. Cui et al. [17] proposed a secure firmware upgrade protocol for Hewlett–Packard (HP) printers. They listed vulnerable firmware versions and searched for printers on the Internet. Then, they checked the firmware versions of the identified printers to determine whether they were vulnerable. Many printers were vulnerable because although HP released patched firmware, most printer users did not upgrade the firmware. For protection, the authors proposed autonomic binary structure randomization, in which the firmware is entered as an input, the unused functions are removed to reduce the surfaces of attack points, and the execution flow of the functions that cannot be disabled by configuration is changed. Their paper proposed a good firmware upgrade protocol; however, it takes a different approach compared to ours. The method proposed in [17] focuses on installing the firmware securely, whereas we focus on transmitting the firmware securely. Furthermore, this system has the management overhead of vulnerable firmware lists.

Zaddach et al. [18] studied firmware analysis. They demonstrated the process of firmware analysis, identification, unpacking, modification, reversal, and packing with usable tools. In addition, they summarized the firmware analysis process.

Previous research can be categorized into two main approaches. One approach proposes effective or economical firmware upgrades [12,16]. However, security issues are not considered or missed. The other approach uses improper assumptions, only suggesting protocols without any experiments [8,14,17].

In contrast, we consider both security and practicality. The firmware upgrade system proposed in this paper also uses existing protocols such as those in [12,16], but we additionally utilize security protocols such as TLS, the SCRAM, and the anonymous Diffie–Hellman protocol to provide secure firmware upgrades. In [14], they assumed a pre-shared key for symmetric encryption, but we utilize the ephemeral Diffie–Hellman key agreement protocol in TLS to agree on an encryption key [19]. Contrary to [8], we consider practical environments and assume that only the firmware server has a pair of public and private keys and correct certification. Furthermore, WAPs do not receive their key from the firmware server. Finally, to determine the differences in the upgrade times according to the environment, we carried out experiments under different conditions.

3. Firmware Upgrade Using Password-Based Authentication

The requirements of the proposed system are based on the basic security network model [20], which ensures the following four security requirements: confidentiality refers to whether an authorized person can access the firmware, authentication refers to whether the upgrade occurs between the correct nodes, integrity refers to whether the system can check that the transmitted data have not been modified, and freshness refers to whether the system can check that the firmware has the correct validation value.

3.1. Network Architecture of the Firmware Upgrade Protocol

Figure 1 shows the network architecture of the firmware upgrade system. The firmware server and WAP are connected via a wired Internet connection. The firmware server stores the firmware of the WAP, and the WAP administrator upgrades the firmware or the upgrade is regularly scheduled. Devices that need to connect to the Internet use the WAP to join it.

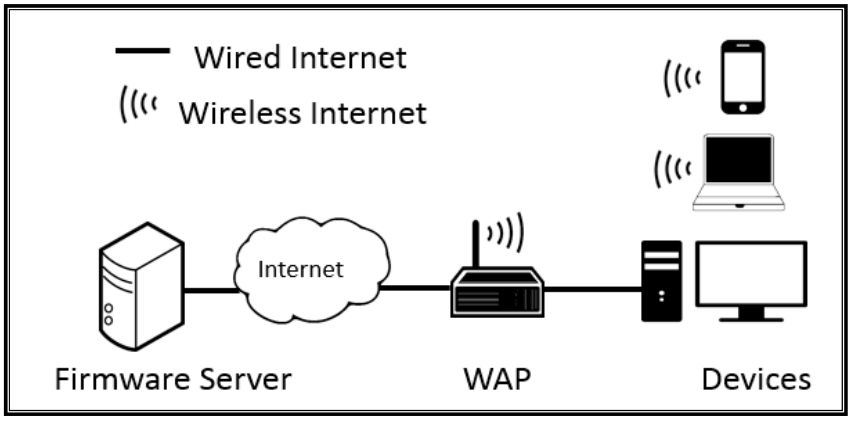

Figure 1. Network architecture of the firmware upgrade system.

In this architecture, we focus on the firmware upgrade between the firmware server and the WAP, which are connected by the Internet. The firmware server and WAP have a sufficient network bandwidth because they use a wired network infrastructure. In addition, the firmware server is generally a desktop PC; thus, it has sufficient performance to upgrade the firmware on the server side. In contrast, the WAP has limited resources.

Because of the difference in performance, we use hybrid cryptography. Exponential computations occur only when needed; the other computations are based on ADD and XOR operations. TLS has successfully implemented this cryptography model [19]. In the proposed protocol, exponential computations occur in the verification of the firmware server certification, the ephemeral Diffie–Hellman key agreement protocol to exchange the key, and the anonymous Diffie–Hellman key agreement protocol to agree on the freshness. In contrast, the firmware upgrade between the WAP and the devices takes place in a different environment. Network interfaces vary, and devices have significant performance differences. Furthermore, some devices are mobile. The model for this environment will be considered in future work.

3.2. Practical Assumptions

The secure firmware upgrade protocol has three assumptions, which are considered to be practical and sustainable for real-world firmware upgrades.

(i) The WAP stores an authentication table that includes the ID, secure hash value, salt, and iteration count of the initial firmware. Furthermore, the password corresponding to the ID is stored in

the firmware server authentication table. This information is used for authentication in the SCRAM process to determine whether the firmware server is the correct one.

This is a very reasonable assumption because the WAP manufacturer can manage the WAP manufacturing process so that the authentication table can be stored in it. In addition, manufacturers can provide a firmware server to support firmware upgrades; thus, they can also store the authentication information. Figure 2 shows the authentication tables of the firmware server and WAP. Figure 2a shows the authentication table in the server, where the ID is used to find the matching password. Figure 2b shows the authentication table in the WAP. It contains the ID, secure hash value, salt, and iteration count. If the password is stored in plaintext in the WAP, it is vulnerable to data exposure. Hence, the WAP stores the password as a hash value. Furthermore, the salt and iteration count are used to secure the hash value. The salt is added to the password, and the result is passed through the hash function several times (equal to the iteration count). This makes it difficult for an attacker to guess the password using the exposed hash value [21,22]. It is a good method for protecting data from exposure, as a WAP is exposed to various attacks via software, the network, and hardware [23].

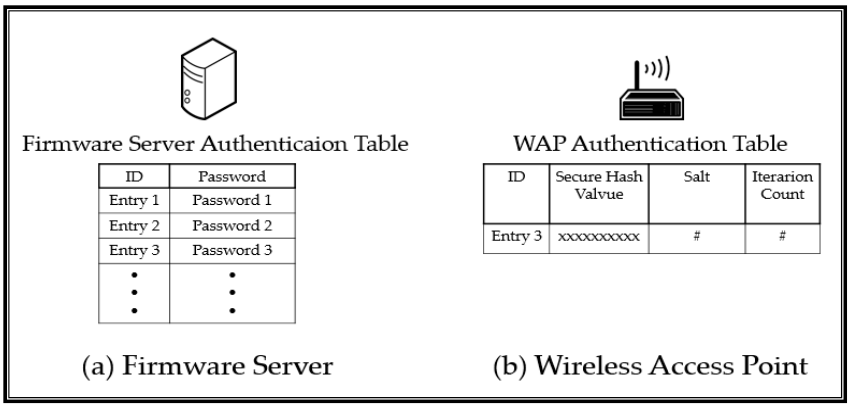

Figure 2. Authentication tables of the (**a**) firmware server and (**b**) WAP.

(ii) The firmware server has a pair of public and private keys and a correct certificate. Generally, asymmetric cryptography is used to solve the key management problem and provide secure services (via authentication, digital signatures, and nonrepudiation). However, this process needs a certificate to verify the public key [24]. If both the firmware server and WAP have a pair of public and private keys, a secure firmware upgrade can easily be achieved. However, it is very difficult to manage and renew a certificate. All new WAPs need to have a correct certificate issued by a certificate authority. For WAPs that have been sold, the user of the WAP should manage any problems with the certificates. In addition, for unsold WAPs, the WAP company should manage the same problems. Furthermore, asymmetric cryptography is unsuitable for use in WAPs because of their limited resources.

Hence, we make practical assumptions by considering the above situation; only the firmware server has a pair of public and private keys and a correct certificate. It is very easy to manage the certificate of a single firmware server. The WAP checks whether or not the firmware server is correct by verifying its certificate.

(iii) The original firmware server is fully secure. A firmware server is verified with a certificate of its own. However, if the firmware server is infected with malware or a virus, it might not support firmware upgrades, or the certificate could be forged. An attacker could impersonate the firmware server by using a forged certificate and then transmit malicious firmware [25,26]. Furthermore, if the authentication table in the firmware server is exposed, the WAP cannot trust the SCRAM process. Hence, we assume that the firmware server is fully secure. This is similar to the assumption in PKI, where the certificate authority is a trustable facility. In the case of a WAP, the authentication table

is secured from data exposure because the password in the authentication table is stored in a hash value [21,22]. A hash value is practically noninvertible.

3.3. Proposed Firmware Upgrade Protocol

Figure 3 illustrates the protocol of the proposed firmware upgrade, which has four phases. In the first phase, the WAP verifies the firmware server certificate and establishes an encrypted channel using TLS. In the second phase, the WAP authenticates the firmware server using the SCRAM, in which the authentication table information consists of an ID, a password, a secure hash value, the salt, and an iteration count. The authentication in the TLS protocol is based on the certificate, but in the SCRAM, it is based on the password. In the third phase, a key is agreed upon using the anonymous Diffie–Hellman key agreement protocol. The key is used to determine the freshness. The authentication and the encrypted channel of the TLS process protects against MITM attacks [10,11]. In the fourth phase, the firmware server adds the freshness to the firmware and transmits it to the WAP. The WAP compares the freshness of the firmware with its own freshness, which was calculated in the third phase. If the freshness is correct, the WAP upgrades its firmware with the received firmware upgrade.

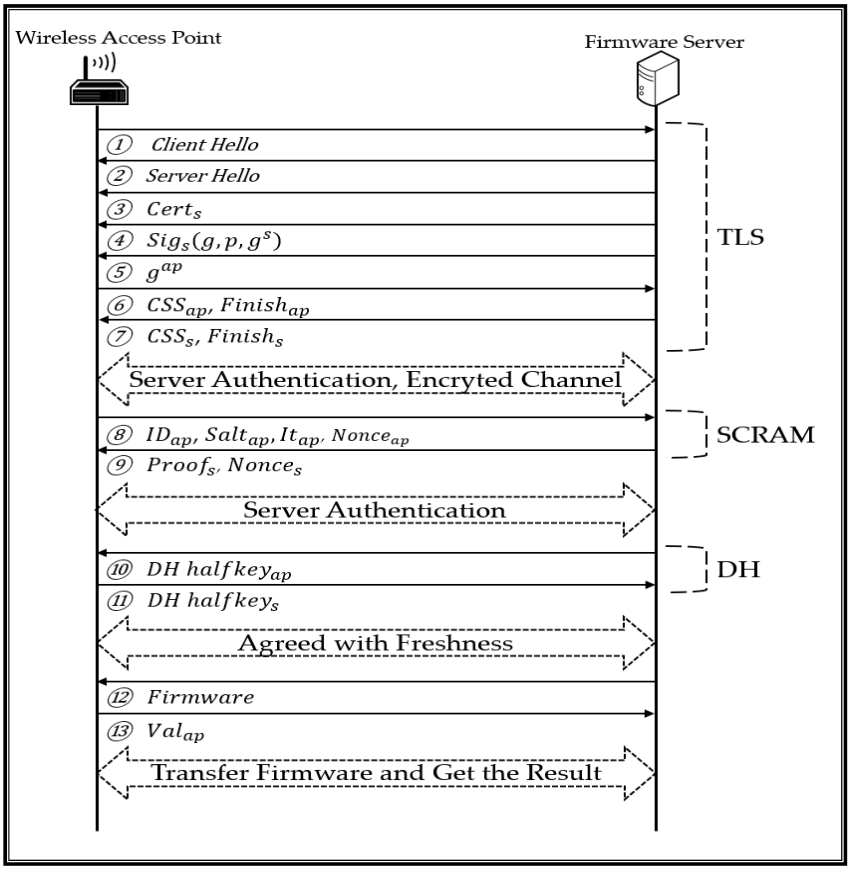

Figure 3. Firmware upgrade protocol.

We considered the ideal assumptions of previous works [8,12], i.e., that a pre-shared symmetric key exists and all servers and WAPs have a pair of public and private keys, from a practical point of view. A symmetric key is calculated by the ephemeral Diffie–Hellman key agreement protocol in the TLS protocol. Only the firmware server has a pair of private and public keys. Furthermore, no secret information is calculated by the firmware server and transmitted to the WAP in plaintext. The authentication information is stored in the WAP and firmware server when the initial firmware is installed. This is appropriate because the WAP manufacturer can manage the manufacturing process

of the WAP and firmware server. Furthermore, the information stored in a WAP is secured from data exposure because it is stored in a secure hash value.

The key calculated by the anonymous Diffie–Hellman key agreement protocol ensures that the calculated values of the firmware freshness are different each time and for each firmware; thus, if a firmware freshness value is revealed, the others remain secure. Consequently, the proposed protocol provides confidentiality, integrity, authentication, and freshness. Furthermore, it can be easily implemented because it uses existing protocols. The details of each step of the protocol are illustrated in Figure 3. In addition, Table 1 defines the notation used in the proposed protocol.

Table 1. Firmware Upgrade Protocol Notation.

Notation	Definition
V_{ap}	WAP TLS Protocol Version
V_s	Firmware Server TLS Protocol Version
R_{ap}	WAP Random Number
R_s	Firmware Server Random Number
SID_{ap}	WAP Session ID
SID_s	Firmware Server Session ID
CS_{ap}	WAP Cipher Suites
CS_s	Firmware Server Cipher Suites
CM_{ap}	WAP Compression Methods
CM_s	Firmware Server Compression Methods
$Cert_s$	Firmware Server Certificate
$Sig_s\,(g, p, g^s)$	Sign the Diffie–Hellman Parameter and Half-key with the Firmware Server Private Key
g^{ap}	WAP Half-key
CSS_{ap}	WAP CipherSuiteSpec
CSS_s	Firmware Server CipherSuiteSpec
ID_{ap}	WAP ID
$Salt_{ap}$	Salt Value Used for the Secure Hash Value in the WAP
It_{ap}	Iteration Counter Used for the Secure Hash Value in the WAP
$Proof_s$	Hash (Password, Salt, It) $\oplus$ $Nonce_{ap}$ $\oplus$ $Nonce_s$
Val_{ap}	Firmware Server Validation Result of the Revised Firmware
$Nonce_{ap}$	WAP Random Number
$Nonce_s$	Firmware Server Random Number

Steps 1–7: In these steps, the TLS protocol is used. TLS is the security protocol in OSI layer 4. It provides authentication, confidentiality, and integrity [19]. The purpose of using the TLS protocol is to verify the firmware certificate to verify that the server is correct and to create an encrypted channel for confidentiality. An encrypted channel ensures that the following processes (e.g., the SCRAM and anonymous Diffie–Hellman protocol) are secure.

Step 1: WAP $\rightarrow$ S: $M_1 = \{V_{ap}, R_{ap}, SID_{ap}, CS_{ap}, CM_{ap}\}$

The firmware upgrade is started when the WAP administrator performs a firmware upgrade or a regularly scheduled application is started [27]. The WAP sends a client hello message to the server S. The client hello contains the TLS version, a random number of 32 bytes, the session ID, and the supported encryption and compression algorithms.

Step 2: S $\rightarrow$ WAP: $M_2 = \{V_s, R_s, SID_s, CS_s, CM_s\}$

When S receives a client hello from the WAP, it selects a TLS version, the encryption algorithms, and a compression method that are compatible with the WAP. Then, S sends a server hello message to the WAP. The server hello contains the selected TLS version, the encryption algorithms, the compression algorithms, a random number of 32 bytes, and the session ID.

Step 3: S $\rightarrow$ WAP: $M_3 = \{Cert_s\}$

After sending M_2 to the WAP, S also sends its own certificate [24]. The TLS protocol defines the four exchange mechanisms in steps 3–4. Of them, we use the ephemeral Diffie–Hellman protocol [19]

because it is more secure than the other mechanisms and provides perfect forward secrecy [28]. We assume that only the firmware server has a pair of public and private keys and a correct certificate in advance. The WAP verifies the certificate using the public key of the firmware server.

Step 4: S → WAP: $M_4 = \{Sig_s\,(g, p, g^s)\}$

After sending M_3 to the WAP, S generates the ephemeral Diffie–Hellman parameter and half-key. Then, S sends this information to the WAP, signed with its private key.

Step 5: WAP → S: $M_5 = \{E_{Puks}\,(Pre - Master\ secret_{ap})\}$

When WAP receives M_4 from S, it verifies the received certificate and obtains the parameter and half-key of the server using firmware server's public key. If the certificate is verified, the WAP proceeds with the firmware upgrade; otherwise, the firmware upgrade is stopped. In this step, the WAP has no certificate; instead, it only sends the half-key.

Step 6: WAP → S: $M_6 = \{CSS_{ap}, Finish_{ap}\}$

After sending M_5 to S, the WAP sends the ChangeCipherSpec, which contains the parameters changed from the pending state to the active state. It also sends the finish message, which contains the verification value of the sent and received messages to S.

Step 7: S → WAP: $M_7 = \{CSS_s, Finish_s\}$

When S receives M_6 from the WAP, it also sends the ChangeCipherSpec, which contains the parameters changed from the pending state to the active state. It also sends the finish message, which contains the verification value of the data sent and received from the WAP.

The WAP authenticates S and establishes an encrypted connection between them using the above processes. It uses the ephemeral Diffie–Hellman protocol to agree on the key because it is more secure than other mechanisms [28,29]. Exponential computations, which take a large amount of computing resources, occur only when the WAP verifies the server certificate and performs the ephemeral Diffie–Hellman key exchange. The encrypted channel is based on symmetric cryptography.

Steps 8–9: In these steps, password-based authentication is performed. It uses the SCRAM and the authentication tables in both S and the WAP. The transmitted values in these steps are secure because the previous process ensures the confidentiality of the channel.

Step 8: WAP → S: $M_8 = \{ID_{ap}, Salt_{ap}, It_{ap}, Nonce_{ap}\}$

When the WAP receives M_7 from S, it generates a nonce, which is random number, and sends the ID, secure hash value, salt, iteration count in the authentication table and the nonce to S.

Step 9: S → WAP: $M_9 = \{Proof_s, Nonce_s\}$

When S receives M_8 from the WAP, it finds the password that matches the received ID and calculates the secure hash value using the received salt and iteration count. In addition, S computes $Proof_s$, which is calculated by an exclusive add operation with the received nonce, its own nonce, and the secure hash value. Then, S sends $Proof_s$ and its own nonce to the WAP.

Password-based authentication is performed using the above processes. Passwords are stored when the initial firmware is installed. Hence, an attacker cannot know them. Furthermore, transmitted values such as the ID, salt, nonce, secure hash value, and iteration count are secure because they are encrypted using the symmetric encryption algorithms of the TLS protocol. Furthermore, if the data of the WAP are exposed by the vulnerabilities of the WAP, the password is secure because it is stored in a secure hash value. Hence, an attacker will find it difficult to guess the password.

Steps 10–11: In these steps, a freshness value is agreed upon using the anonymous Diffie–Hellman protocol [29]. Its known vulnerability, MITM attacks, is prevented by authentication and the encrypted channel of the TLS.

Step 10: WAP → S: $M_{10} = \{DH\ halfkey_{ap}\}$

When the WAP receives M_{10} from S, it compares the received proof with its own proof. If they are equal, the WAP generates a half-key for the anonymous Diffie–Hellman protocol and sends it to S. If not, the upgrade is stopped.

Step 11: S → WAP: $M_{10} = \{DH\ halfkey_s\}$

When S receives M_{10} from the WAP, it also generates a half-key and sends it to the WAP.

The WAP and S agree on a key using the anonymous Diffie–Hellman protocol. The key is used for the firmware freshness. An MITM attack is prevented by ensuring the confidentiality of the channel.

Steps 12–13: In these steps, the firmware is transmitted. In the previous processes, authentication, channel encryption, and freshness agreement are achieved for the secure firmware upgrade. Hence, the transmission is secure.

Step 12: S $\rightarrow$ WAP: $M_{12} = \{Firmware\}$

After sending M_{11} to the WAP, S adds the freshness to the firmware and sends it to the WAP.

Step 13: WAP $\rightarrow$ S: $M_{13} = \{Val_{ap}\}$

When the WAP receives M_{12} from S, it compares the freshness of the received firmware with the agreed freshness. If they are equal, the system proceeds with the upgrade and sends a validation message; otherwise, the upgrade is terminated.

4. Implementation and Experiments

4.1. Prototype Implementation

We implemented a prototype for the proposed firmware upgrade. A desktop PC acted as the firmware server, and a Raspberry Pi B+ acted as the WAP because it has limited resources. Multiple Raspberry Pi 2 B devices were used as additional WAPs to measure the performance using the upgrade time according to the number of WAPs. Table 2 summarizes the specifications of the desktop, Raspberry Pi B+, Raspberry Pi 2 B, Wi-Fi dongle, and router. Figure 4 shows the software architecture of the firmware upgrade system, and Appendix A describes the flow of the firmware upgrade program in the WAP and firmware server. In addition, Figures 5 and 6 show photographs of the prototypes. The WAPs consist of one Raspberry Pi B+ and three Raspberry Pi 2 B devices. The WAPs are located on the left sides of Figures 5 and 6, and the router next to it provides the wired or wireless Internet connection. The firmware server was located in a different network area and has a public IP address. The network architecture of the experiment is the same as that described in Section 3.1.

We monitored the states of the Raspberry Pi B+ and Raspberry Pi 2 B and controlled them using an external terminal and USB interfaces.

Table 2. Specifications of the devices.

Devices	Specifications
Desktop PC	Ubuntu 16.04 LTS 64 bits, 10 GB RAM, Intel core i5 2.80 GHz × 4
Raspberry Pi B+	Raspbian, 700 MHz single-core ARM1176JZF-S, 256 MB RAM
Raspberry Pi 2 B	Raspbian, 900 MHz quad-core ARM Cortex-A7, 1 GB RAM
Router	PISnet P150N, RT5350, 150 Mbps/170 Mbps 802.11b/g/n
Wi-Fi dongle	Iptime N100mini, 150 Mbps 802.11b/g/n
Wi-Fi dongle	NEXT-201N MINI, 150 Mbps 802.11b/g/n

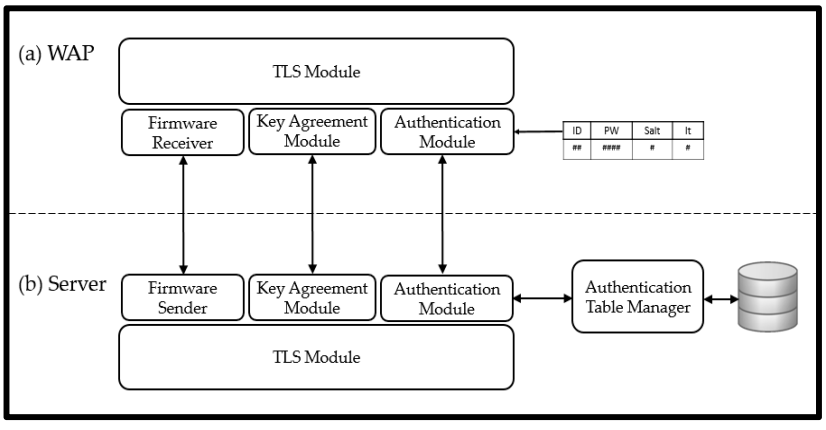

Figure 4. Software architecture of the firmware upgrade: (**a**) WAP side and (**b**) server side.

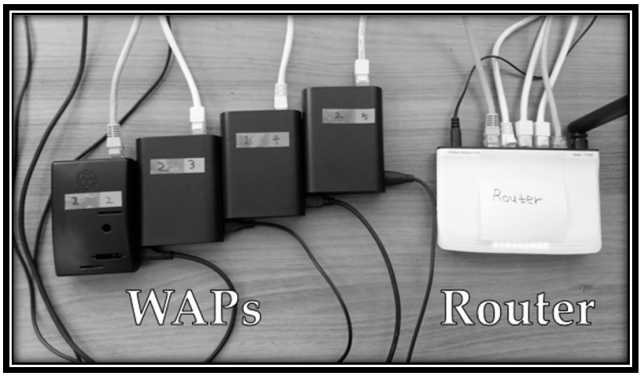

Figure 5. WAPs connected to the Internet via a wired router.

Figure 6. WAPs connected to the Internet via a wireless router.

4.2. Performance of the Firmware Upgrade System

The size of a WAP's firmware generally varies from 2 MB to 10 MB, as determined from the firmware sizes of the top 10 most popular WAPs of 2015 [30]. We performed 10 tests for each firmware size in a wired connected environment and reported the averages of the upgrade time and resource usage. Table 3 summarizes the results of the experiments. The upgrade time increases in proportion to the size. The other values remain constant regardless of the firmware size. VIRT represents the virtual memory usage, RES represents the shared memory usage, and MEM shows the percentage of consumed memory. Figure 7 compares the desktop PC and Raspberry Pi B+ file I/O waiting for the firmware upgrade. In fact, the difference in the performance of the desktop PC and Raspberry Pi B+ is caused by several different aspects. Hence, they cannot simply be compared. However, we can identify that the file I/O waiting (10.12%) for the Raspberry Pi B+ is much higher than the file I/O waiting (0.16%) for the desktop PC. The Raspberry Pi B+ uses an external microSD card for storage, causing more file I/O waiting than other storage technologies, e.g., an HDD, would incur. Therefore, we can expect that the firmware upgrade time will decrease for a WAP that uses another storage technology.

Table 3. Results of the firmware upgrade.

Firmware Size (MB)	Time (s)	VIRT	RES	SHR	MEM (%)
2	9.63	17,080	11,920	6256	2.7
4	18.94	17,080	11,964	6300	2.6
6	28.15	17,080	12,036	6372	2.7
8	37.67	17,080	11,989	6287	2.7
10	46.89	17,080	12,010	6303	2.7

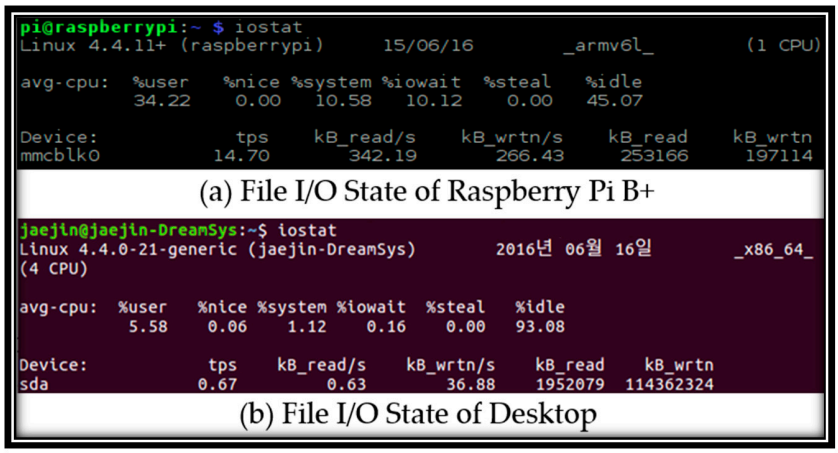

Figure 7. Comparison of the file I/O states.

The authentication table in the WAP consists of a 3-digit ID, a 64-digit secure hash value using SHA2, a 64-digit salt, and a 3-digit iteration count. Hence, only 134 bytes (3 bytes + 64 bytes + 64 bytes + 3 bytes) are needed to store the authentication table. This is a very small overhead in the WAP.

The software size of the firmware upgrade in the WAP is 138,712 bytes. In the case of small WAPs, the code would need to be optimized because they do not have sufficient space to store the firmware upgrade software.

As shown in Figure 8, the upgrade time of the Raspberry Pi 2 B devices are two times faster than Raspberry Pi B+ devices. The firmware size is 2 MB. Figures 9 and 10 show the upgrade times according to the number of WAPs in the wired and wireless connected environments. Basically, there is one Raspberry Pi B+, and Raspberry Pi 2 B devices are used as additional WAPs. In the wired connected environment, the number of WAPs does not affect the firmware upgrade time. However, in the wireless connected environment, the upgrade time of the Raspberry B+ slowly increases according to the number of WAPs. Commonly, wireless communications are severely affected by interference. Thus, the firmware server and WAP cannot perform a firmware upgrade because there is more interference when there are more WAPs.

Figure 11 shows the percentage increase in the upgrade time for the wireless connected environment relative to those of the wired environment. Generally, all WAPs require more time than those in the wired connected environment. In the case of the Raspberry Pi B+, the upgrade time increases as WAPs are added.

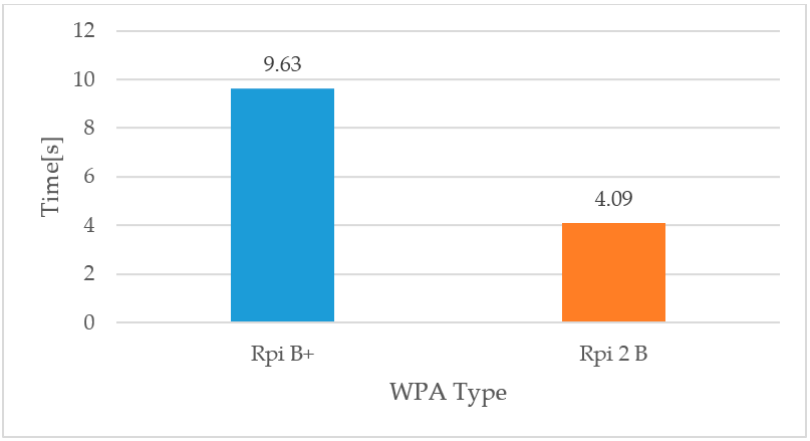

Figure 8. Comparison of the upgrade time for different devices.

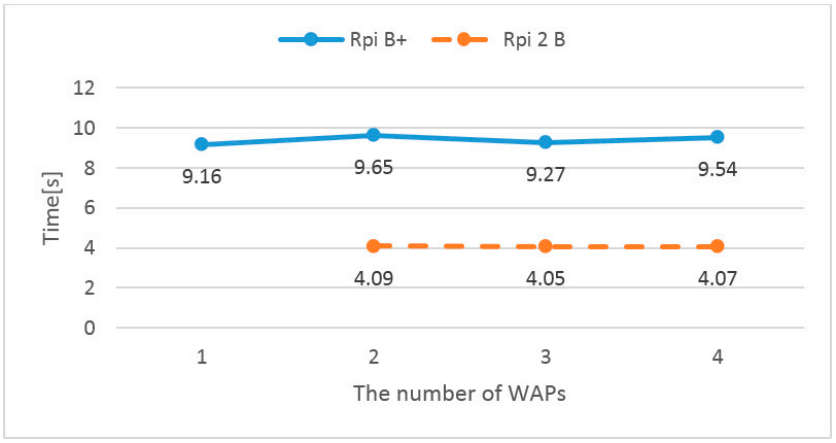

Figure 9. Upgrade times in the wired connected environment.

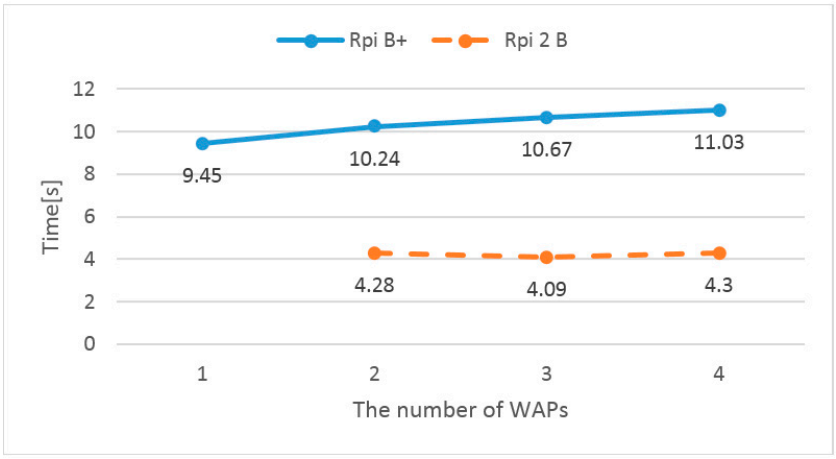

Figure 10. Upgrade times in the wireless connected environment.

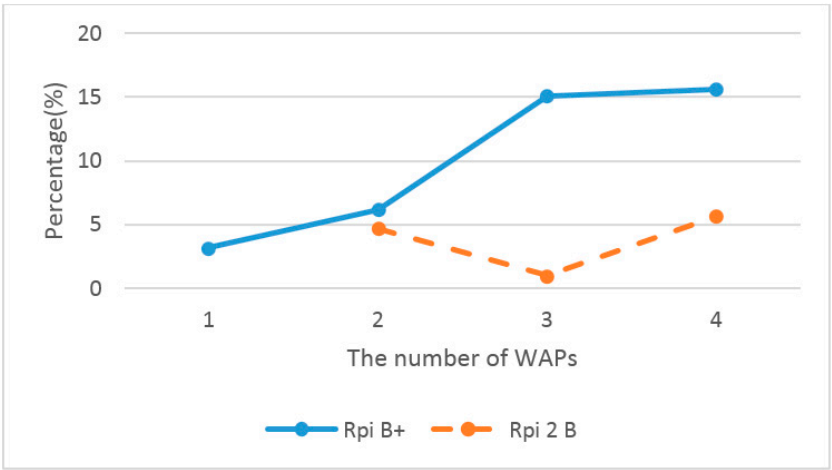

Figure 11. Increase in the upgrade times for the wireless connected environment compared to those of the wired connected environment.

4.3. Security Analysis of the Protocol

4.3.1. Security Analysis

Confidentiality—Confidentiality is provided if a third party cannot access to the data transmitted between WAPs and the firmware server. To ensure confidentiality, cryptography is used. The encryption algorithms of the proposed protocol is based on symmetric-key algorithm; the proposed protocol used AES-256 [31]. This algorithm provides strong encryption for data. Therefore, the proposed protocol provides confidentiality.

Entity authentication—Entity authentication ensures that an identified party is involved in a protocol at a given instant [32]. Proposed protocol has two-step authentication. First, the firmware server sends a X.509 certificate. Then, the WAP receives the certificate and verifies it using public key of the server. Second, authentication is processed based on pre-installed authentication table. The WAP has the hash value of password not password itself. The WAP sends the hash value of password, salt, nonce and ID. The server lookups the matched password with ID and then calculates the proof value and transmits the proof value. Then the WAP verifies the proof. Attackers cannot know the pre-install value. Therefore, the second authentication is secure. In addition, the hash value is more secure because the password passes a hash function as many as iteration counters, and salt is added to password before hashing. Therefore, proposed protocol provides entity authentication.

Nonrepudiation—Nonrepudiation is provided when the WAP or the firmware server cannot repute the sender or the receiver of data [15]. To provide nonrepudiation, the protocol should use asymmetric-key algorithms. However, proposed protocol used symmetric-key algorithms. Therefore proposed protocol does not provided nonrepudiation.

Integrity—Integrity means that changes need to be done only by authorized entities and through authorized mechanisms [33]. Proposed protocol's message has MAC (Message Authentication Code) for integrity. Therefore proposed protocol provides integrity.

Key freshness—Key freshness is provided when the session key is a new value not the session key used [33]. The firmware server and the WAP always generate random number for a session key. Therefore proposed protocol ensures key freshness.

4.3.2. Analysis of the Proposed Protocol's Security Attributions

Known session key security—Known session key security is provided although a session key is compromised, other session keys are secure [34]. The session key is generated by random values and hash functions. Accordingly, every session key is independent. Consequently, if one session key is compromised, an attacker cannot access to other session and cannot guess other session keys. Therefore, proposed protocol provides known session key security.

Forward secrecy—Forward secrecy is provided if one long-term key is compromised, past session keys are not compromised. Proposed protocol used ephemeral key for a session key. Therefore, proposed protocol provides forward secrecy. In addition, it is proven that the protocol using ephemeral Diffie–Hellman algorithm provides perfect forward secrecy [28].

Key control—Key control is provided if the session key is derived from both the information of the firmware server and the WAP. In addition, other entity cannot affect the process of driving session key [35]. As we described in Known session key security, the session key is generated by random values of the firmware server and WAP, and the third-party does not participate in session key driving. Therefore, proposed protocol provided key control.

Unknown key-share resilience—Unknown key share attack may occur when the connection between a firmware server and a WAP is established, but the WAP or the server believes it is sharing the key related to connection with another entity [36,37]. Proposed protocol's message always have a MAC (Message Authentication Code). Inside the MAC, MAC write secret is contained. It has authentication key for the outbound message. Therefore, the firmware server and the client can verify the identity of message. Consequently, proposed protocol provides Unknown key share resilience [38,39].

Key-compromise impersonation resilience—Key-compromise impersonation attacks are the attacks in which the adversary exploits the knowledge of the long-term private key of the firmware server or the WAP to impersonate any entity [40]. Proposed protocol has two-step entity authentication. An attacker should have a certificate and an authentication table to impersonate other entity. To impersonate a WAP, an attacker needs the authentication table because client side does not have a certificate in proposed protocol. Although an attacker, who had the authentication table of a WAP, impersonates other entity, he cannot adversely affect the firmware server or other WAPs. Only the firmware server can send the firmware to the WAP unilaterally, the attacker does not respond to the messages received from WAPs. The firmware server received the validation message to check the installation of transmitted firmware, it does not store or react for them [38,40].

Firmware authentication—The goal of proposed protocol is the firmware upgrade. Therefore it is important to check whether the firmware received from the firmware server is legitimate or not. Proposed protocol provides the authentication value to validate the firmware. At each process in firmware upgrade, the authentication value is newly generated by Diffie–Hellman protocol. Accordingly, authentication is different in each firmware. Therefore, an attacker cannot install malicious firmware to client and cannot guess the authentication value of the other firmware. Consequently, proposed firmware protocol provides authentication for firmware.

4.3.3. Limitations

In the ephemeral Diffie–Hellman protocol in TLS, there is a slight difference from its common use. Generally, the client (WAP) also has a certificate. The ephemeral Diffie–Hellman parameter and half-key are signed with the server's private key, but the WAP sends a half-key in plaintext to the server. Even if the half-key is transmitted in plaintext, this is secure because attackers do not know the private key of the server [15]. In addition, they cannot perform an MITM attack because they do not have a forged server certificate. However, server vulnerability could result in a forged firmware server certificate, which would be dangerous [25,26,41]. Hence, securely managing the firmware server is essential.

5. Conclusions

In this paper, we proposed a sustainable and practical firmware upgrade for a WAP. As the number of ICT devices increases, WAPs will become more important. However, attackers can install malicious firmware by exploiting the vulnerabilities of the firmware upgrade process to decrease the network bandwidth or obtain the traffic including private data such as search histories, login information, and device usage patterns. There are several related works that address this problem, but they all have conflicts between security and practicality. However, the proposed firmware upgrade satisfies both security and practicality and utilizes security protocols such as TLS, the SCRAM, and the anonymous Diffie–Hellman protocol to provide confidentiality, integrity, authentication, and freshness. In addition, the firmware upgrade system can be easily implemented because it uses existing security protocols. We considered real-world environments to design a sustainable and practical firmware upgrade protocol. Only the firmware server has a pair of public and private keys and a certificate. Exponential computations occur only when necessary for verification of the server's certificate and key exchange for the WAP, which has limited resources. We implemented a prototype of the upgrade protocol and performed experiments to analyze the security of the proposed protocol. The results of the experiments show that the resource usage and code size are reasonable for WAPs. However, the upgrade time was not measured properly because the file I/O waiting was very high (10.12%) because of the use of an external microSD card for storage. In addition, we checked that the system provides confidentiality, integrity, authentication, and freshness by analyzing the security of the protocol. Furthermore, the security property of the protocol and perfect forward secrecy are ensured. In addition, the protocol protects against known session key attacks, key control and unknown key-share attacks.

The possible problems caused by the absence of the WAP's certificate remain as future work. In addition, adaptable firmware upgrades in WAPs and devices in home networks, which have different environments that include mobility as well as different resources and network interfaces, are also planned for future work.

Author Contributions: All the authors contributed equally to this work. All authors read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Firmware Upgrade Pseudocode

```
<Firmware Server pseudocode>
BEGIN;
Receive Client Hello;
Send Server Hello;
Send Certificate;
Send Ephemeral DH Parameters and DH Half-key signed by own private key;
If Connection is terminated
Stop Upgrade;
Receive WAP Ephemeral DH Half-key
Send CipherSuiteSpec and finish
Receive CipherSuiteSpec and finish
Receive ID, Salt, Iteration Count, Nonce
Compute Proof and send
If Connection is terminated
Stop Upgrade;
Send Anonymous DH Parameters and DH Half-key;
Receive Anonymous DH Half-key
Compute key for Freshness
Add Freshness to Firmware and send
Receive Validation message
END;
```

```
<WAP pseudocode>
BEGIN;
Send Client Hello;
Receive Server Hello;
Receive Certificate;
Receive Ephemeral DH Parameters and DH Half-key;
If Certificate is incorrect,
Stop Upgrade;
Get the Ephemeral DH Parameter and DH Half-key using Server Private Key
Send Client Ephemeral DH Half-key
Receive CipherSuiteSpec and finish
Send CipherSuiteSpec and finish
Send ID, Salt, Iteration Count, Nonce
Receive Proof
If Proof is incorrect
Stop Upgrade;
Receive Anonymous DH Parameters and DH Half-key;
Send Anonymous DH Half-key
Compute key for Freshness
Receive Firmware
If Freshness is incorrect
Stop Upgrade;
Send Validation message and process the Upgrade
END;
```

References

1. Egevang, K.; Francis, P. The IP Netw ork Address Translator (NAT). Available online: http://www.rfc-editor.org/info/rfc1631 (accessed on 23 August 2016).
2. Gartner. Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016, Up 30 Percent from 2015. Available online: http://www.gartner.com/newsroom/id/3165317 (accessed on 23 August 2016).
3. Proofpoint. *Proofpoint Uncovers Internet of Things (IoT) Cyberattack*; Proofpoint Release: Sunnyvale, CA, USA, 2014.
4. Stiawan, D.; Idris, M.Y.; Abdullah, A.H. Penetration Testing and Network Auditing: Linux. *JIPS* **2015**, *11*, 104–115.
5. Nohl, K.; Lell, J. BadUSB—On Accessories that Turn Evil. Available online: http://www.slideshare.net/cisoplatform7/sr-labs-badusbv2 (accessed on 23 August 2016).
6. Delugré, G. Closer to Metal: Reverse-Engineering the Broadcom NetExtreme's Firmware. Available online: http://esec-lab.sogeti.com/static/publications/10-hack.lu-nicreverse_slides.pdf (accessed on 23 August 2016).
7. Miller, C. Battery Firmware Hacking. Available online: http://www.hakim.ws/BHUS2011/materials/Miller/BH_US_11_Miller_Battery_Firmware_Public_WP.pdf (accessed on 23 August 2016).
8. Choi, B.C.; Lee, S.H.; Na, J.C.; Lee, J.H. Secure firmware validation and update for consumer devices in home networking. *IEEE Trans. Consum. Electron.* **2016**, *62*, 39–44. [CrossRef]
9. Tom, S. *The Hackers New Weapons Routers and Printers*; MIT Technology Review: San Diego, CA, USA, 2015.
10. Dacosta, I.; Ahamad, M.; Traynor, P. Trust No One Else: Detecting MITM Attacks against SSL/TLS without Third-Parties. In *Computer Security—ESORICS 2012*; Springer: Berlin, Germany, 2012; pp. 199–216.
11. Karapanos, N.; Capkun, S. On the Effective Prevention of TLS Man-in-the-Middle Attacks in Web Applications. In Proceedings of the 23rd USENIX Security Symposium (USENIX Security 14) 2014, San Diego, CA, USA, 20–22 August 2014; pp. 671–686.
12. Odat, H.A.; Nsour, A.; Ganesan, S. Firmware over the Air Ad-Hoc Network, FOTANET. In Proceedings of the 2015 IEEE International Conference on Electro/Information Technology (EIT), Dekalb, IL, USA, 21–23 May 2015; pp. 101–106.
13. Postel, J. RFC 793: Transmission Control Protocol. Available online: https://tools.ietf.org/html/rfc793 (accessed on 23 August 2016).
14. Tsague, H.D.; Van Der Merwe, J.; Moabalobelo, T. Secure Firmware Updates for Point of Sale Terminals. In Proceedings of the 10th International Conference on Cyber Warfare and Security, Kruger National Park, South Africa, 24–25 March 2015; Academic Conferences Limited: Sonning Common, UK, 2015; p. 337.
15. Forouzan, B.A.; Mukhopadhyay, D. *Cryptography and Network Security (Sie.)*; McGraw-Hill Education: New York, NY, USA, 2011; pp. 2–7, 507–548.
16. Steger, M.; Karner, M.; Hillebrand, J.; Rom, W.; Armengaud, E.; Hansson, M.; Boano, C.A.; Römer, K. Applicability of IEEE 802.11s for Automotive Wireless Software Updates. In Proceedings of the 13th International Conference on Telecommunications (ConTEL) 2015, Graz, Austria, 13–15 July 2015; pp. 1–8.
17. Cui, A.; Costello, M.; Stolfo, S.J. When Firmware Modifications Attack: A Case Study of Embedded Exploitation. Available online: http://ids.cs.columbia.edu/sites/default/files/ndss-2013.pdf (accessed on 23 August 2016).
18. Zaddach, J.; Costin, A. Embedded Devices Security and Firmware Reverse Engineering. Available online: https://media.blackhat.com/us-13/US-13-Zaddach-Workshop-on-Embedded-Devices-Security-and-Firmware-Reverse-Engineering-WP.pdf (accessed on 23 August 2016).
19. Dierks, T. The Transport Layer Security (TLS) Protocol Version 1.2. Available online: https://www.ietf.org/rfc/rfc5246.txt (accessed on 23 August 2016).
20. Stallings, W. *Cryptography and Network Security, 4/E*; Pearson Education India: Kormangala, India, 2006; pp. 22–24.
21. Newman, C.; Menon-Sen, A.; Melnikov, A.; Williams, N. RFC 5802: Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms. Available online: https://tools.ietf.org/html/rfc5802 (accessed on 23 August 2016).

22. Turan, M.S.; Barker, E.; Burr, W.; Chen, L. Recommendation for Password-Based Key Derivation. Available online: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf (accessed on 23 August 2016).
23. Moein, S.; Gebali, F.; Traore, I. Analyzis of covert hardware attacks. *J. Converg.* **2014**, *5*, 26–30.
24. Cooper, D. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. Available online: https://tools.ietf.org/html/rfc5280 (accessed on 23 August 2016).
25. Burkholder, P. SSL Man-in-the-Middle Attacks. Available online: http://www.windowsecurity.com/articles-tutorials/authentication_and_encryption/Understanding-Man-in-the-Middle-Attacks-ARP-Part4.html (accessed on 23 August 2016).
26. Ellison, C.; Schneier, B. Ten risks of PKI: What you're not being told about public key infrastructure. *Comput. Secur. J.* **2000**, *16*, 1–7.
27. Fagan, M.; Khan, M.M.H.; Nguyen, N. How does this message make you feel? A study of user perspectives on software update/warning message design. *Hum. Centric Comput. Inf. Sci.* **2015**, *5*, 1–26. [CrossRef]
28. Sheffer, Y.; Holz, R.; Saint-Andre, P. Recommendations for Secure Use of Transport. Layer Security (TLS) and Datagram Transport. Layer Security (DTLS). Available online: https://tools.ietf.org/html/rfc7525 (accessed on 23 August 2016).
29. Rescorla, E. Diffie–Hellman Key Agreement Method. Available online: https://www.ietf.org/rfc/rfc2631.txt (accessed on 23 August 2016).
30. Top 20 Rated Wireless Access Points Reviews 2016. Available online: https://www.vbestreviews.com/top-10-rated-wireless-access-point-reviews-2015/ (accessed on 23 August 2016).
31. FIPS. 197: The official AES standard. Figure 2: Working scheme with four LFSRs and their IV generation LFSR1. *LFSR* **2001**, *2*, 1–3.
32. Bellare, M.; Rogaway, P. Entity Authentication and Key Distribution. In Proceedings of the Annual International Cryptology Conference, Barbara, CA, USA, 22–26 August 1993; Springer: Berlin, Germany, 1993; pp. 232–249.
33. Paar, C.; Pelzl, J. *Understanding Cryptography: A Textbook for Students and Practitioners*; Springer Science & Business Media: Berlin, Germany, 2009; pp. 319–329, pp. 331–353.
34. Stinson, D.R. *Cryptography: Theory and Practice*; CRC Press: Boca Raton, FL, USA, 2005; pp. 436–438.
35. Mitchell, C.J.; Ward, M.; Wilson, P. Key control in key agreement protocols. *Electron. Lett.* **1998**, *34*, 980–981. [CrossRef]
36. Akram, R.N.; Markantonakis, K.; Mayes, K. A Privacy Preserving Application Acquisition Protocol. In Proceedings of the 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, Liverpool, UK, 25–27 June 2012; pp. 383–392.
37. Blake-Wilson, S.; Menezes, A. *Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol*; Public Key Cryptography, Springer: Berlin, Germany, 1999; pp. 154–170.
38. Cremers, C.; Horvat, M. Improving the ISO/IEC 11770 Standard for Key Management Techniques. In Proceedings of the International Conference on Research in Security Standardisation, London, UK, 16–17 December 2014; pp. 215–235.
39. Chen, L.; Mitchell, C. Security Standardization Research. In Proceedings of the First International Conference SSR 2014, London, UK, 16–17 December 2014.
40. Hlauschek, C.; Gruber, M.; Fankhauser, F.; Schanes, C. Prying Open Pandora's Box: KCI Attacks against TLS. In Proceedings of the 9th USENIX Workshop on Offensive Technologies (WOOT 15), Washington, DC, USA, 24 June 2015.
41. Holz, R.; Riedmaier, T.; Kammenhuber, N.; Carle, G. X.509 Forensics: Detecting and Localizing the SSL/TLS Men-in-the-Middle. In *Computer Security—Esorics 2012*; Springer: Berlin, Germany, 2012; pp. 217–234.