Security Analysis of Certified Wireless Universal Serial Bus Protocol

Rishabh Dudheria and Wade Trappe

Abstract— **The Certified Wireless Universal Serial Bus (USB) protocol is designed to leverage the existing USB infrastructure in the wireless environment to provide high-speed communication over short distances. We analyze this protocol with a concentration on its security features and identify a number of shortcomings and attacks on the protocol. We have also proposed enhancements to the design of the current protocol to improve its security.**

Index Terms—**Wireless University Serial Bus protocol, Security analysis, USB**

I. Introduction

THE WIRELESS Universal Serial Bus (USB) protocol [1] is designed to provide robust high-speed wireless connectivity by utilizing the WiMedia MB-OFDM Ultra-wideband (UWB) radio platform [2], [3]. It aims to deliver speeds of 480 Mbps at 3 meters to 110 Mbps at 10 meters.

Security mechanisms for the Wireless USB protocol are implemented on top of the USB core specification. The USB control channel is used to handle all the security operations to make the specification media independent. The goal of Wireless USB security is to provide the same level of protection as the Wired USB 2.0 [4] standard.

The Wireless USB protocol allows each host to connect to a maximum of 127 devices. A group consisting of a Wireless USB host and its associated devices is referred to as a Wireless USB cluster. The Wireless USB standard aims to keep the data exchanged between the host and device private and protected in the wireless environment, along with protecting the owners/user's equipment from an attacker. In order to meet these goals, it provides mutual authentication between the host and the devices during connection setup so that the participating entities have the opportunity to validate each other.

In this paper, we analyze the Wireless USB protocol with an emphasis on its security, list possible attacks, enumerate the points that lack clarity and suggest changes to enhance the robustness of the protocol.

The rest of this paper is organized as follows. Section II provides a brief overview of the security features of the Wireless USB protocol (a detailed description of the protocol is beyond the scope of this paper). Section III lists a collection of possible attacks on the protocol. Section IV lists additional shortcomings of the protocol. Section V proposes changes to the protocol to make it more robust and secure. Finally, Section VI provides a concluding discussion.

R. Dudheria and W. Trappe are with the Electrical and Computer Engineering Department and Wireless Information Network Laboratory at Rutgers University, North Brunswick, NJ 08902 USA (email: {rishabh, trappe}@winlab.rutgers.edu).

Pad bits	Tail bits	FCS	Payload	Tail bits	HCS	MAC header	PHY header	PLCP preamble

Fig. 1. Wireless USB packet format

MIC (8)	Application Payload	WUSB Hdr (2)	SFN (6)	Encrypt Offset (2)	Rsrvd (1)	TKID (3)

Fig. 2. Modified payload for encrypted packet.

II. Overview Of Wireless USB Protocol

A. Packet Format

The packet formats used in the Wireless USB protocol are defined in the corresponding Medium Access Control (MAC) layer standard [3]. The general packet format used in the protocol is shown in Fig. 1. It consists of a physical layer (PHY) preamble, PHY header, MAC header and a data payload. An encrypted packet consists of a modified payload as shown in Fig. 2. The security related fields in the packet are Temporal Key ID (TKID), Reserved (Rsrvd), Encryption Offset, Secure Frame Number (SFN), and Message Integrity Code (MIC). These fields are present in a packet only if the Security bit component of the Frame Control field in the MAC header is set to 1.

There are four basic packet types used in this protocol:

1) Micro-scheduled Management Command (MMC) packet: These are cluster broadcast control packets transmitted by the host using secure packet encapsulation with the Encryption Offset field in the Security Header set to the length of the MMC payload. The host uses the Group Key (that is shared by all the members of the corresponding cluster) to generate the MIC for these packets, which provides authentication of the packet. In particular, the MMC contain Information Elements (IE), which are a part of the information and control mechanisms for the Wireless USB channel.

2) Protocol data packet: This is transmitted either by the host or a device using secure packet encapsulation in most cases such that the entire body of the application payload is encrypted.

3) Protocol handshake packet: These packets can only be transmitted by the device to the host. When secure packet encapsulation is present, the entire packet is sent in plaintext with its integrity being protected by the MIC.

4) Device notification packet: These are sent by the device to the host mostly using secure packet encapsulation in such a way that the entire packet is transmitted in plaintext with its integrity being protected by the MIC.

B. *Encryption*

The standard specifies the use of AES-128 [5] Counter mode with CBC-MAC (CCM) as the symmetric encryption algorithm to provide integrity and encryption. The CCM nonce provides uniqueness to each message. The MIC is calculated using the counter-mode blocks. The message and the MIC are encrypted using the keystream provided by the encryption blocks.

C. *Keys*

The specification defines two types of keys, namely the Master Key and Session Keys.

1) Master Key: It is the long lived key, which is used as a shared secret for authentication and derivation of session keys.

a) Connection Key (CK): The CK is the primary 128- bit key used for establishing connections. Each device and host share a unique CK, which is setup during the initial connection.

2) Session Keys: These are the short lived keys used for encryption and decryption. These keys last only during the lifetime of a connection, i.e., they are discarded when a connection ends.

a) Pair-wise Temporal Key (PTK): The PTK is a 128- bit key used for the encryption and decryption of data packets between the host and the device. It is derived during a 4-way handshake process (refer to Fig. 4). Again, each device should have a unique PTK.

b) Group Temporal Key (GTK): The GTK is a temporal key shared by all the members of the Wireless USB cluster. The host uses the GTK to send secured broadcast messages to all the devices in its cluster. The corresponding devices use the GTK to verify the authenticity (by verifying the MIC) of the broadcast messages received from the host. The standard specifies that the devices may not use the GTK for encryption. However, the standard imposes no requirements on GTK generation.

c) Key Confirmation Key (KCK): The KCK is a 128-bit key used for providing message integrity during authentication. It is derived during authentication and discarded upon completion of authentication.

d) Key Derivation Key (KDK): A KDK (256-bit key) is computed if an additional key is required by any other applications.

Temporal Key ID (TKID): The TKID is a 3 byte ID, which is used by the MAC layer as the name for the GTKs and PTKs. The key used to encrypt a secured packet is identified using the TKID. The TKID values are created by the host and given to the devices at the time of key derivation or key distribution. The specification does not mention how these ID's should be generated by the host.

D. *Secure Relationship*

The protocol defines a Connection Context (CC) as a secure relationship between a host and a device. The CC needs to be stored in the non-volatile memory at both the host and device end for future reconnections. It consists of 3 pieces of information.

1) Connection Host ID (CHID): The CHID is a unique 128-bit host ID. The CHID is used by the device to locate the host.

2) Connection Device ID (CDID): The CDID is a unique 128-bit device ID, which is assigned by the host to a device during the connection process.

3) CK: The CK is a 128-bit key used to establish connections using this context. This key can be periodically updated by the host.

E. *Association Models*

Association is defined as the process of establishing the first time connection between the hosts and the devices. The specification mentions four different ways to perform association viz., 'Cable association', 'Numeric association', 'Fixed PIN association', and 'Near Field Communication (NFC) association'. We were unable to find a detailed description of the Fixed PIN association and Near Field Communication (NFC) association models. We hereby provide a short description of the Cable association model and the Numeric association model based on [6]. The devices with a wired USB interface are prescribed to implement the Cable association model, while the devices with a display must implement the Numeric association model. The hosts are supposed to support as many association models as possible.

1) Cable association model: In this model, the user connects the device to the host using a USB cable. The CC is then delivered by the host to the device. The host and the device then perform a 4-way handshake for mutual authentication and to derive the session keys.

2) Numeric association model: In this model, the CK is derived at both ends using Diffie-Hellman (DH) Key exchange [7]. The various steps in the exchange have been shown in Fig. 3 at an abstract level with only the information necessary to understand the model. D represents the device and H represents the host; PK_D and PK_H represent the device and the host public key respectively; SHA-256 is the Secure Hash Standard [8]; DHKey is the secret Diffie-Hellman key; N_D is the number of digits that the device can display; V is the verification number; *Trun-128{}* denotes truncation of all, but the first 128 bits of the argument; and HMAC-SHA-256 represents the Keyed-Hash Message Authentication Code [9] that uses SHA-256 as the hash function.

The device and host each generate a random secret of 256 bits, say A and B respectively. This is used to compute the Diffie-Hellman public key (of size 384 byte), $PK_D = g^A$ (mod p) and $PK_H = g^B$ (mod p). The device then sends a hash commitment of its public key concatenated with the number of digits it can display to the host. This is done to avoid man- in-the-middle attacks. The host then sends its public key to the device. Consequently, the device returns its public key and the number of digits it can display. The host computes the hash of the device's public key concatenated with the number of digits it can display and verifies it with the earlier hash commitment to ensure that the values match; otherwise the association is aborted. After this step, the host and the device independently compute the DHKey. To further protect against a man-in-the-middle attack, the protocol requires that both the host and the device independently calculate a verification number and display 2 to 4 digits of the same so that the user can validate that the association has been completed successfully. Once this is done, both the sides compute the CK. The host then delivers the CHID and CDID to the device. A KDK is then computed independently at both the ends if a key is required by any other application. Finally, all the temporary values computed or generated during this process are erased except the CC and the KDK (if any).

1. D→H: SHA-256 (PK_D || N_D)
2. H→D: PK_H
3. D→H: PK_D and N_D
4. H computes SHA-256 (PK_D || N_D)
5. D computes DHKey = SHA-256 ($PK_H{}^A$ mod p)
6. H computes DHKey = SHA-256 ($PK_D{}^B$ mod p)
7. Both H and D compute
   V= SHA-256 (PK_D||PK_H||"displayed digest")
8. Both H and D compute
   CK= Trun-128{HMAC-SHA-256_{DHKey}("connection key")}
9. H→D: CHID, CDID
10. If needed, both H and D compute
    KDK= HMAC-SHA-256_{DHKey}("key derivation key")

Fig. 3. Numeric association model.

F. 4-Way Handshake

This serves the dual purpose of mutual authentication and session key derivation. The messages exchanged during this process are shown at an abstract level in Fig. 4 with only the information necessary to understand the process.

$H_{N\ once}$ and $D_{N\ once}$ are 128-bit random nonce generated by the host and device respectively; $MIC_{KCK()}$ represents the MIC calculated for the contents inside the bracket with fresh Key Confirmation Key (KCK). New keys are derived from the shared secret CK through a Pseudo Random Function with output length X (PRF-X) as follows:

Key Stream = PRF-256 (CK, Host DevAddr, Device DevAddr, TKID, "Pairwise keys", $H_{N\ once}$ ||$D_{N\ once}$, 32), which is split to form the initial management and data keys. The least significant 16 bytes and the most significant 16 bytes of the Key Stream become the KCK (which is discarded after authentication) and PTK respectively.

The host initiates the 4-way handshake by sending a TKID and H_N once to the device during phase 1. The device then uses this information to generate the KCK and the PTK. During phase 2, the device provides the host with the TKID, D_N once and a MIC calculated over the entire packet payload using the KCK. The host then derives the corresponding keys and verifies the MIC provided by the device. Upon successful verification, the host has proof that the device holds the correct CK. Consequently, it proceeds to phase 3 to provide proof to the device that it holds the correct CK by sending a message containing the TKID, $H_{N\ once}$ and a MIC computed over the entire packet payload with the KCK. The device computes the corresponding MIC and verifies it with the MIC sent by the host to ensure that the values match, stores the session key, and informs the host that it has successfully installed the session key during phase 4; otherwise it aborts the connection and removes the derived session key.

Phase 1
H→D: TKID, H_{Nonce}
Phase 2
D→H: TKID, D_{Nonce} , MIC_{KCK}(TKID,CDID, D_{Nonce})
Phase 3
H→D: TKID, H_{Nonce} , MIC_{KCK}(TKID,CDID, H_{Nonce})
Phase 4
D→H: Successfully installed session key

Fig. 4. 4-way handshake.

After the successful completion of the 4-way handshake, the host provides the device with the current GTK, which is sent in a secured manner using the PTK.

G. Replay Prevention

The protocol defines three 48 bit components: a Secure Frame Counter (SFC), SFN and a replay counter (RC) to avoid replay of packets sent using the GTK and the PTK. The host and the device each maintain a separate SFC to encrypt transmitted packets, and a separate RC to decrypt received packets for each session key. Initially, these counters are set to a value of zero when a new key is installed. The SFN is an image of the SFC being used to encrypt the packet.

The SFC associated with the key is incremented when a packet is encrypted for transmission and is then copied to the SFN field of the packet. The SFN value for each successive retry of a packet is greater than the last attempt.

The receiver compares the packet SFN value with the value of the RC associated with the decryption key. A packet is declared to be a replay of a previous packet if the SFN value is less than or equal to the value of RC (and is consequently discarded); otherwise the RC is set to be equal to the SFN of the received packet.

A. Key Management

Hosts are responsible for all the key management operations. The PTK can expire if it is used to encrypt 2^{48} packets. In such a case, the host performs an additional 4-way handshake with the device to derive a new PTK. The GTK is changed whenever a device leaves the Wireless USB cluster. Since, synchronization of the new GTK is difficult, the protocol requires each device to be capable of holding two GTKs. The host distributes the GTKs in numerical order based on a 4-bit index value. The low bit of the session key index is used as the table index so that Key_2 can replace Key_0, Key_3 can replace Key_1 , and so on. The host installs and begins to use a new GTK only after the last device in the cluster has confirmed receiving it. The device can discard an older GTK only after the host begins to use the new GTK. The host must not use the older GTK once it starts using the new GTK. However, the specification does not mention what should be done if the same GTK is used to encrypt more than 2^{48} packets.

III. Threat Analysis of Wireless USB Protocol

We now look at several attacks that may be conducted against the Wireless USB protocol. To begin with, we assume an attack model in which an attacker can eavesdrop on every message, replay stored messages, and insert forged messages; but it does not know the shared CK and PTK between the host and an uncompromised device. We further assume that it is also possible for an attacker to intercept and block delivery of a message. This assumption is further supported by the fact that the messages tend to experience higher loss rates in a wireless medium.

A. *Message Spoofing*

A device that has successfully established a connection with the host and has acquired the GTK can masquerade as the host and send forged MMC packets to the cluster containing the following Information Elements (IEs), which may lead to a denial of service (DoS) attack:

1) WHOST DISCONNECT IE: To inform all the cluster members that they are being disconnected resulting in spoofed disassociation message.

2) WCHANNEL STOP IE: To notify cluster members that the Wireless USB channel is being stopped.

3) WCHCHANGEANNOUNCE IE: To notify cluster members that the host is moving the Wireless USB channel to a different PHY channel.

B. *Sybil Attack*

A successful connection between a device and a host requires the CC (consisting of 16 bytes of CHID, CHID, and CK each) to be stored in the non-volatile memory at both the device and host end. A malicious device can make 127 connections to the same host if it has enough memory ($16 \times 3 \times 127$ bytes ≈ 6 KB). This can exhaust all the resources of the host and thus, prevent any legitimate devices from accessing the corresponding host leading to a Sybil attack [10].

C. *Water Torture Attack*

In order to save power, the host can go in sleep state, while enabling the devices to perform remote wakeup. A malicious device can prevent the host from sleeping, effectively draining off its power.

D. *Packet Dropping*

An adversary can change the SFN associated with the group key in a packet from its original value to a much higher value, and then recalculate the MIC associated with the corresponding packet. This causes legitimate packets with lower SFN values to be dropped by the devices.

IV. Short Comings/Lack of Clarity

The protocol does not impose any requirements on CK generation in the Cable association model, and when new CKs are distributed by the host using the 'Set Connection Context' command. The specification requires the host to distribute a unique CK to every device. This means that the CK should be generated randomly using a uniform probability distribution on the space of 128-bit strings. The standard should specify this explicitly.

The protocol also fails to define how the GTK and the TKID are generated by the host. We were also unable to locate the size of the GTK in the specification. Furthermore, the standard also fails to mention how the KDK would be generated by the host and the device if the initial association were to take place using the Cable association model.

The specification defines the CCM nonce to be made up of the SFN, TKID, DestAddr and SrcAddr to ensure its uniqueness. Now, if there are more than 2^{48} packets encrypted with the same GTK, then the SFN associated with the corresponding packets would roll over resulting in the same nonce to be used again for encryption. The standard should clearly specify what should be done if the same GTK is used to encrypt more than 2^{48} packets.

The standard specifies that the Set Connection Context command can be used by the host to modify a device's CC. It also mentions that the CC delivered in this way should always be protected. The designer's should have explicitly mentioned that the CC's delivery has to be protected using the PTK; otherwise, a malicious device can announce a change in the physical channel being used by the host and then use this command to achieve a session hijack.

Chapter 7, p. 163 of the specification mentions about some host association keys that are distributed by the host to the devices using the 'Set Key' command. We could not comprehend what these keys referred to as they were clearly not PTKs or GTKs.

The latest version of the standard [1] refers to an 'Association Model Specification' document on p. 141 and 142 of chapter 6, which we were unable to find on the web. The previous version of the Wireless USB specification [11] has a corresponding 'Association Models Supplement' [6] document that was published on March 2, 2006. However, [1] seems to be referring to some other document as it describes a new Fixed PIN association model on p. 142 of chapter 6, which has not been described in [6].

V. ENHANCEMENTS

A. A. Elliptic Curve Diffie-Hellman (ECDH) Exchange

The designers have used 3072 bit ephemeral DH key for initial connection in the Numeric association model, which provides a cryptographic strength of 128 bits. We are proposing the use of ephemeral ECDH keys for computing CK in the Numeric association model as it can provide the same cryptographic strength with a lower key size of 283 bits. This smaller key size can save a lot or resources such as memory, computing power and battery [12]. 'Association Models Supplement FAQ' [13] mentions that the Menezes-Qu-Vanstone (MQV) cryptographic method was considered for Numeric association, but was rejected because it is covered by patents and requires excessive computation. ECDH on the other hand is non-proprietary. There are several standards available from IEEE, IETF, etc., providing guidelines regarding its practical deployment.

A brief review of ECDH is provided here as a detailed description of this algorithm is beyond the scope of this paper. The curve parameters are fixed in advance for all the hosts and devices designed to use the Wireless USB protocol. Initially, both the device and host create their private keys by generating a random integer. They calculate their public key by multiplying their private key with the generating point. Then, they exchange their public keys over the wireless medium. The shared secret is generated at each end by multiplying the public key received from the other end with the local private key. Let the device and host generate the random private keys A and B respectively. The device and host then calculate their public keys X and Y as follows: X = A.G and Y = B.G, where G is the basepoint on the curve. The public keys X and Y are then exchanged over the wireless channel. The device and host then derive the shared secret by computing A.Y and B.X respectively.

Thus, the same steps as described in the Numeric association model can be implemented by using the ephemeral ECDH keys instead of the ephemeral DH keys.

B. CK Generation

Another source of weakness in the current protocol is that the host generates the CK in the Cable association model as well as when new CKs are distributed using the Set Connection Context command on its own. This principle implies that a device must trust that the host always generates a new CK that is cryptographically separated from all the other CKs generated by all the hosts. Further, if the CKs are generated by the host using a random number generator then it must be perfect; otherwise, if it exhibits some bias then it can expose the CK and hence the PTK. A safer way to compute the CK would be to calculate it using the bits contributed by both the host and the device; for example, CK = HMAC-SHA (host generated connection key, some random value generated by the device). This is similar to the weakness associated with the generation of authorization key in 802.16 [14].

VI. DISCUSSION

The Wireless USB protocol employs information that is sent in the clear, but what saves the protocol from an outsider attack is the use of out-of-band mechanisms (USB cable, user) to validate the association. For instance, in the Numeric association model any entity can capture the public keys and hence display the correct digits required for verification, but it still cannot cause any harm to the participants as the user needs to validate the association. Our paper has pointed out many shortcomings in the current specification of the Wireless USB protocol. These should be fixed by the designers as soon as possible to avoid ambiguity once the products enter the market. We have also outlined a number of ways to enhance the security of the current protocol: the most important of them being the use of ephemeral ECDH keys instead of ephemeral DH keys for initial association.

REFERENCES

[1] Wireless Universal Serial Bus Specification, Wireless USB Promoter Group Std., Rev. 1.1, Sep. 2010. [Online]. Available: http://www.usb.org/developers/wusb/docs

[2] Multiband OFDM Physical Layer Specification, WiMedia Alliance Std., Rev. 1.2, Feb. 2007.

[3] Distributed Medium Access Control (MAC) for Wireless Networks, WiMedia Alliance Std., Rev. 1.2, Mar. 2008.

[4] Universal Serial Bus Specification, Universal Serial Bus Implementers Forum (USBIF) Std., Rev. 2.0, Apr. 2000, including all published errata. [Online]. Available: http://www.usb.org/developers/docs/

[5] "FIPS PUB 197, Advanced Encryption Standard," National Institute of Standards and Technology, Nov. 2001. [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[6] Association Models Supplement to the Certified Wireless Universal Serial Bus Specification, Wireless USB Promoter Group Std., Rev. 1.0, Mar. 2006. [Online]. Available: http://read.pudn.com/downloads162/sourcecode/unix linux/network/736567/wusb 2007 0214/WUSB AM Spec r10.pdf

[7] E. Rescoria, "Diffie-Hellman Key Agreement Method," RFC 2631, Jun. 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2631.txt

[8] "FIPS PUB 180-2, Secure Hash Standard," National Institute of Standards and Technology, Aug. 2002. [Online]. Available: http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

[9] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC:keyed-hashing for message authentication," RFC 2104, Feb. 1997. [Online]. Available: http://www.faqs.org/rfcs/rfc2104.html

[10] J. R. Douceur, "The Sybil Attack," in Revised Papers from the First International Workshop on Peer-to-Peer Systems, ser. IPTPS '01. London, UK, UK: Springer-Verlag, 2002, pp. 251–260.

[11] Wireless Universal Serial Bus Specification, Wireless USB Promoter Group Std., Rev. 1.0, May 2005. [Online]. Available: http://www.usb.org/wusb/docs/WirelessUSBSpecification r10.pdf

[12] K. Lauter, "The advantages of elliptic curve cryptography for wireless security," Wireless Communications, IEEE, vol. 11, no. 1, pp. 62–67, Feb. 2004.

[13] "Association Models Supplement to the Certified Wireless Universal Serial Bus Specification Frequently Asked Questions," Wireless USB Promoter Group, Jun. 2007. [Online]. Available: http://www.usb.org/developers/wusb/WUSB AM FAQ 2007 06 19.pdf

[14] D. Johnston and J. Walker, "Overview of IEEE 802.16 security," Security Privacy, IEEE, vol. 2, no. 3, pp. 40–48, May-June 2004.