Secure Software Upload in an Intelligent Vehicle via Wireless Communication Links

Syed Masud Mahmud, *Member, IEEE*, Shobhit Shanker, *Student Member, IEEE*
Irina Hossain, *Student Member, IEEE*

***Abstract*— The demand for drive-by-wire, telematics, entertainment, multimedia, pre-crash warning, highway guidance, remote diagnostic, etc. will significantly increase the complexity of a vehicle's software modules. From time to time, the vehicle's software may need to be updated due to many reasons such as the introduction of new features in vehicles, changing the navigation map, fixing software bugs, etc. Software updates must be done in secure modes to avoid any future disasters due to malfunctions of the vehicle. In this paper, we propose an architecture for secure software uploads in vehicles. We provide a detailed description of the secure software upload process.**

***Key words*— Software update, security, key management, intelligent vehicles.**

I. Introduction

About half a century ago a vehicle was mostly a mechanical device. Today, a significant part of a vehicle's manufacturing cost goes towards the implementation of electronic components. The demand for drive-by-wire, telematics, entertainment, multimedia, pre-crash warning, highway guidance, remote diagnostic, etc. will significantly increase the complexity of in-vehicle communication networks, hardware and software modules. From time-to-time, it will be necessary to upgrade vehicles' software modules. For example, the map of a vehicle's navigation system may need to be updated when new roads, houses and offices are built. A company may need to monitor its newly designed vehicles, via wireless communication links, to determine the performance of the vehicles in terms of fuel efficiency, emission and other driving conditions. The company can then improve, if necessary, the performance of its vehicles by remotely adjusting some software parameters in the vehicles' electronic modules. Updating software modules in the future intelligent vehicles, on a regular basis, will be required to keep the vehicles compatible with the infrastructure of the intelligent transportation system.

Remote software upload, via wireless communication links, has many advantages over upload at a service station. Some of the advantages are as follows:

- The consumers do not need to take their vehicles to service stations. Thus, remote software upload operations will save consumers' valuable time.
- The personnel at the service stations do not need to spend time on vehicles on an individual basis eliminating labor costs from the auto manufacturers as well as from the consumers.
- The auto manufacturers can immediately fix bugs or upload new features in software modules without being delayed for a long period of time, which may save a significant amount of money from legal costs.

To upload software in vehicles, it is critically important that this be done in a secure environment. Otherwise, the system would be susceptible to security attacks, which will compromise its safety and functionality. This paper presents a mechanism for secure software upload in an intelligent vehicle. The scope of this paper is software upload for one vehicle only. Thus, if the same software needs to be uploaded in multiple vehicles, then the upload operation must be repeated for each vehicle one at a time.

The rest of the paper is organized as follows. Section II presents some basic materials on secure communications. Section III describes our architecture. Section IV shows system requirements, and Section V presents the conclusions.

II. SOME BASIC MATERIALS

During the last several years, interest in using the wireless communication technologies have grown significantly. Bluetooth features are becoming very common in cell phones, PDAs, laptops, etc. More and more homes are getting broadband connections and using Wi-Fi technology for wireless in-home networking. The

Manuscript received December 15, 2004.

Syed Masud Mahmud is with the Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI 48202 USA (Phone: 313-577-3855; fax: 313-577-1101; e-mail: smahmud@eng.wayne.edu).

Shobhit Shanker is with the Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI 48202 USA (e-mail: sshanker@wayne.edu).

Irina Hossain is with the Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI 48202 USA. (e-mail: ihossain@wayne.edu)

0-7803-8961-1/05/$20.00 ©2005

automotive industry has also started introducing wireless technology, such as Bluetooth, for its in-vehicle networking [1]. If secure wireless networks can be implemented at reasonable costs, then consumers' demand for in-vehicle wireless networks may increase significantly. The government may also mandate auto companies to have wireless networks in vehicles so that the vehicles can communicate with each other to issue pre-crash warnings; and law-enforcing authorities can monitor vehicles' emission, speed and other items related to violations of traffic laws.

Auto companies and suppliers must use secure wireless links to upload software in their vehicles. Otherwise, hackers may tamper with software code during the uploading process, leading to possible disasters during the operation of the vehicles. The current techniques that are used for setting up a secure communication link are mainly client-server based techniques such as SSL and VPN. Brief descriptions of these techniques are shown below.

A. SSL (Secure Socket Layer)

The SSL protocol is the Web standard for encrypting communications between users and SSL (secure socket layer) enabled e-commerce sites [2]. The SSL layer runs on the top of TCP/IP layer. The SSL security protocol provides data encryption, server authentication, message integrity, and optional client authentication for a TCP/IP connection. The SSL protocol includes two sub protocols [3]: the SSL record protocol and the SSL handshake protocol. The SSL record protocol defines the format used to transmit the data. The SSL handshake protocol involves using the SSL record protocol to exchange a series of messages between an SSL enabled server and SSL enabled client when they first establish a connection. This exchange of messages is designed to facilitate the actions like: authentication of the server to the client, authentication of the client to the server, selection of cryptographic algorithms that they both support, use of public key encryption techniques to generate shared secrets, etc.

B. VPN (Virtual private network)

VPN is a private network that uses a public network (usually the Internet) to connect remote sites or users together [4]. Instead of using a dedicated, real-world connection such as a leased line, a VPN uses "virtual" connections routed through the Internet from the company's private network to the remote site or employee. VPN mainly uses IPSec (internet protocol and security) protocol, which in turn uses the tunneling method for making data packets immune to attacks. IPSec can encrypt data between various devices, such as router-to-router, firewall-to-router, PC-to-router, PC-to-server, etc. Tunneling is the process of placing an entire packet within another packet and sending it over a network. The network and both parties understand the protocol of the outer packet, called the tunnel interface.

C. Symmetric and Public Key Encryption

There are mainly two types of encryption techniques: the symmetric-key and public-key encryptions [5]. In the symmetric-key encryption, both parties use the same key to encrypt and decrypt messages. In this encryption technique, both parties must have the same key in their system before they are going to exchange messages. The public-key encryption uses a combination of a private key and a public key. A computer (say X) can send its public key to other computers with which it wants to establish communications, but it will securely keep its own private key. A message that is encrypted by the public key of X, can only be decrypted by X using its private key. The encryption/decryption process using public/private key mechanism is very time-consuming compared to that using symmetric key mechanism. Normally, two parties use the public/private key technique to exchange their symmetric key. For example, if another computer (say Y), which has the public key of X, wants to establish a secure communication link with X, then Y generates a symmetric key and sends it to X after encrypting it using the public key of X. X then uses its private key to decrypt the message sent by Y, and collects the symmetric key from the decrypted message. After that, X and Y communicate using the symmetric key generated by Y.

III. THE PROPOSED ARCHITECTURE FOR SECURE SOFTWARE UPLOAD

Here we present and discuss the architecture for secure software upload in vehicles' electronic modules. We assume that all the vehicles are equipped with wireless interface units to communicate with the infrastructure of the Intelligent Transportation System (ITS). A company's server can communicate with a selected group of vehicles via either the cellular network or the ITS infrastructure. Since the cell phone towers cover almost all the road systems of the country, the same tower can also be used as the Intelligent Transportation Tower (ITT).

A. Authentication Keys

We assume that at the time of manufacturing a vehicle, a set of authentication keys is installed in the vehicle. The same set of keys is also kept in a central server. Every time a vehicle is going to be authenticated by an ITT, the ITT gets the set of keys of the vehicle from the central server, and uses one key to authenticate the vehicle. Different authentication keys are used to authenticate the vehicle at different times. When all the keys of the set are used for authenticating the vehicle, a new set of keys is generated and sent to the vehicle. The new set of keys is also kept in the central server. Successive authentications of the vehicle are done using the new set of keys. After authenticating the

vehicle, the ITT or the central server issues a symmetric key to the vehicle. The ITT and/or the central server can securely communicate with the vehicle using the symmetric key.

B. Key Management for Software Upload

Assume that automotive company X wants to upload software in one of its vehicles (say Vehicle V). Let Y be the supplier of the software. X generates a set of link keys (P_1, P_2, . . , P_N) and sends it to Y using a secure link such as SSL or VPN. Note that the link keys will be used to establish secure links between Supplier Y and the target Vehicle V. X then securely sends the same set of link keys to Vehicle V using either the cellular network or the infrastructure of ITS. Now, both Vehicle V and the Supplier Y have the same set of link keys. The supplier and the vehicle can establish a secure link using one of these link keys, say P_1. Other link keys will be used to establish future secure links between the vehicle and the supplier. Since a given link key is not going to be used more than once, both the supplier and the vehicle delete the key P_1 from the set of link keys after they have established a secure link for the first time. After that, the supplier and the vehicle generate some symmetric encryption keys. Using these encryption keys, the supplier can send encrypted software to the vehicle. Figure 1 shows the key transfer and software upload process. Since each vehicle has its own set of keys, neither the vehicle nor the software vendor can send any unauthorized software to other vehicles without the consent of the automotive company.

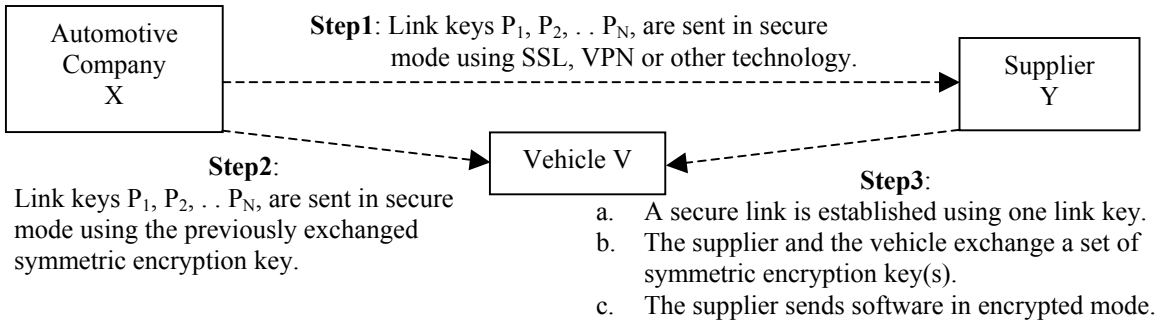

Figure 1: Key transfer and software upload process.

C. Sending Multiple Copies of the Software can Significantly Improve the Security Level

To increase the security level of the software upload process, we recommend that the supplier should send at least two copies of the software to the vehicle. These two or more copies of the software must be sent after some random time intervals. First, the supplier will establish a secure link with the vehicle using one of the link keys, and then send a copy of the software. After a random time interval, the supplier will again establish a secure link with the vehicle using another link key, and send the second copy of the software.

The vehicle must have memory buffers to keep copies of the software. After receiving both copies of the software, the vehicle checks whether the two copies are exactly same. Since the two copies of the software are sent from the supplier to the vehicle at two different times, it is very unlikely that a hacker will be able to change both copies of the software exactly at the same point in the code. Thus, it is very unlikely that the two copies of the software will be exactly same if one or both of them are changed. If the two copies of the software are the same, the vehicle sends a positive acknowledge signal to both X (automotive company) and Y (supplier). If the two copies of the software are not same, then the vehicle asks the supplier to retransmit the unmatched packets.

To avoid hackers from detecting a predicted order of packet transmission, the packets of the software can be sent in some random orders. In that case, even if a hacker can change one packet of the first copy of the software, it will be difficult for the hacker to detect the same packet in the second copy of the software. As a result, it will be more difficult for the hacker to change the same packet in both copies of the software.

After receiving two identical copies of the software, the vehicle initiates a process for replacing the old software of a module by a copy of the new software. The process of replacing the old software is initiated when the vehicle is not in motion and the ignition is off. If the vehicle is in motion, then it may issue a warning to the driver indicating that new software is ready to be loaded in a module. Then the driver may stop the vehicle and turn the ignition off at his/her convenience. The vehicle then keeps a backup copy of the old software in a temporary buffer. After that, the vehicle loads the new software into a module. The backup

copy can be used to undo the upload operation if there are any problems with the vehicle's functionality after the upload operation.

After successfully uploading the software in a vehicle, the automotive company (X) decides whether or not to keep the remaining link keys in the vehicle. If X determines that Y has no more software to upload, then X sends a signal to the vehicle to delete the remaining link keys. This way, the vehicle can keep itself protected from any insider attacks by the employees of Y. In the future, if Y wants to upload another software in the vehicle, then X issues another set of link keys to both Vehicle V and Supplier Y, and the entire process of software upload is repeated.

D. Only One Copy of the Software Appended with a Message Digest (MD).

If only one copy of the software is to be sent, then the security level of the upload process can be increased by appending a message digest (MD) with the software. The supplier can create a 128-bit message digest of the software using the MD5 algorithm [6]. The supplier can then encrypt the message digest and send it along with the software. On the other hand, the vehicle can also create a 128-bit message digest based on the software received by the vehicle. After that, the vehicle can decrypt the message digest sent by the supplier and compare it with its calculated message digest. If both match, then the vehicle can accept the software.

This particular technique, which uses a message digest (MD), has some disadvantages over the other technique that sends at least two copies of the software. If a hacker can successfully change one packet of the software, then the decrypted message digest will not match with the vehicle's calculated message digest. As a result, the supplier will need to retransmit the entire software, because the supplier does not know which particular packet is changed. If a hacker can successfully change a packet of every transmission, then the vehicle will never be able to upload the software.

E. Two Copies of the Software with a Message Digest (MD)

A better way of uploading software in a vehicle would be sending two copies of the software along with the message digest in each copy. If some packets of the first copy (including the message digest) do not match with the corresponding packets of the second copy, then the vehicle asks the supplier to send the unmatched packets. After receiving two identical copies of the software, along with the message digest, the vehicle calculates a message digest based on the software received from the supplier. The vehicle then compares this calculated message digest with the message digest received from the supplier. If they match, the vehicle accepts the software. If they do not match, then the hacker was able to change both copies of some packets (including the message digest). The vehicle then asks the supplier only to send one or more copies of encrypted message digest. The supplier sends the additional copy(copies) of the message digest after some random time interval. After receiving the additional copy of the message digest, the vehicle compares it with the calculated message digest. If they match, the vehicle accepts the software. Otherwise, the vehicle rejects the software. Figure 2 shows the flow diagram of the software upload process where two copies of the software are sent along with the message digest (MD) in each copy.

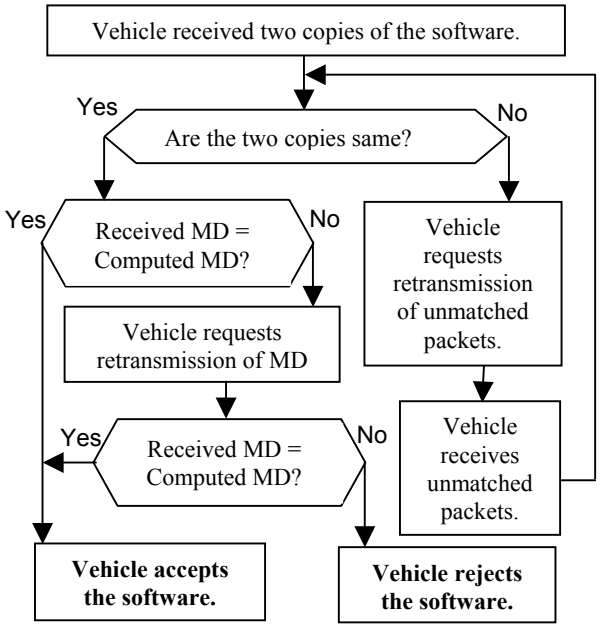

Figure 2: Software upload algorithm.

F. Uploading Software in Multiple Vehicles

Software upload in only one vehicle will be necessary if a particular vehicle has some unique problems with its functionality. Uploading software only in one vehicle is a unicast process. The scope of our current work, presented in this paper, is software upload in one vehicle only. However, if the automotive company wants to add new features to a large number of vehicles, then software upload operations will be required for multiple vehicles. Uploading software in multiple vehicles is a multicast process. Many different multicast algorithms have been designed for mobile devices [7], [8]. Some of these algorithms can be used to extend our work for software upload in multiple vehicles. We intend to do that in our future work.

G. Protection of Keys in the Servers of the Automotive Company and the Software Vendor

All keys must be appropriately protected in the servers of

both the automotive company and the software vendor. However, this is not a unique problem with our work. The same problem exists in all security systems. For example, the systems that use private- and public-key mechanisms must guard the private keys. Whatever techniques and mechanisms are used to guard the private keys of a system, similar techniques and mechanisms could be used to protect the vehicle keys in the servers of the automotive company and the software vendor. The keys must be kept in tamper resistant devices. However, since the scope of this paper does not deal with the protection of keys in servers, we do not want to discuss this issue any further.

H. The Wireless Gateway of a Vehicle

A vehicle must have a wireless gateway for it to receive software via wireless links. This wireless gateway would be like an ECU (electronic control unit) of the vehicle. The wireless gateway should have access to the vehicle's wired bus (e.g. a CAN bus) so that software can be sent to the targeted ECU from the wireless gateway. When the ignition is off, the wireless gateway should be in a low power receive mode to avoid battery draining. Upon detection of a signal for software upload, the wireless gateway will switch itself to the full power mode to communicate with the nearest ITT. After receiving the software the wireless gateway can save the software in its own buffer. Later on, when the ignition is turned on, the software can be transferred to the targeted ECU.

IV. SYSTEM REQUIREMENTS

In this section, we show the requirements of memory size, bandwidth and length of encryption keys for secure software upload operations.

A. Memory Requirement

A vehicle needs two memory buffers: one to keep a backup copy of the current software and another one to keep a copy of the new software. The size of each buffer should be at least equal to the longest code of a vehicle module. Normally, the engine control module contains the longest code. A buffer size of 5 Mbytes should be good enough to hold the software of the engine control module. Thus, the two buffers together need at most 10 Mbytes.

B. Bandwidth Requirement

Unlike vehicle-to-vehicle communication for pre-crash warning that requires high bandwidth, the bandwidth requirement of a vehicle's wireless link for uploading software is not a serious issue. The software doesn't need to be transmitted from the supplier to the vehicle in a short period of time (say within a few minutes). The supplier can take as much time as needed to send the software to the vehicle. Since the transfer process of the two copies of the software from the supplier to the vehicle is transparent to the driver of the vehicle, a very low bandwidth (say 1 Kbyte/sec) link may be good enough. To avoid any bandwidth limitations for the overall system, including the supplier, cellular and ITS infrastructure, the software can be transmitted at off peak hours (say after midnight) when there may be very little traffic from other types of communication services.

C. Size of Encryption Keys

Longer encryption keys need more CPU time to do encryption and decryption than that needed by shorter encryption keys. However, longer keys provide better security than shorter keys. Let us compare the key lengths for two different cases of software upload process to provide the same level of security. In one case, longer keys are used and only one copy of the software is sent from the supplier to the vehicle. In the second case, shorter keys are used, but two copies of the software are sent from the supplier. Let L_1 and L_2 be the length of keys for the first and second case, respectively. The probability that a packet of the first case will be changed is $k2^{-L_1}$, where k indicates how fast the hacker can decrypt information compared to the vehicle. For example, if k=2 then the hacker can decrypt information twice as fast as the vehicle. Let n be the total number of packets in the software. The probability that the vehicle will receive all n packets in clean form is $\left(1-k2^{-L_1}\right)^n$. Then, the probability that at least one packet will be changed by the hacker is

$$p_1 = 1-\left(1-k2^{-L_1}\right)^n \quad (1)$$

Similarly, for the second case, where two copies of the software are sent, the probability that the first copy will be changed is $1-\left(1-k2^{-L_2}\right)^n$. The probability that the second copy of the software will be changed in the same packet, as in the first copy, is $k2^{-L_2}$. Hence, the probability that tampered software will be uploaded in the vehicle for the second case is

$$p_2 = \left(1-\left(1-k2^{-L_2}\right)^n\right)k2^{-L_2} \quad (2)$$

To have the same level of security for both cases, we have to make $p_1 = p_2$, i.e. $1-\left(1-k2^{-L_1}\right)^n = \left(1-\left(1-k2^{-L_2}\right)^n\right)k2^{-L_2}$.

Hence, $nk2^{-L_1} - \frac{n(n-1)}{2}k^2 2^{-2L_1} + \ldots =$

$$\left(nk2^{-L_2} - \frac{n(n-1)}{2}k^2 2^{-2L_2} + \ldots\right)k2^{-L_2} \quad (3)$$

Most of the e-commerce transactions through the internet use 128-bit keys. However, for any reasonable key size

(say 40 bits or longer), the values of $nk2^{-L_1}$ and $nk2^{-L_2}$ are very small. Ignoring the higher order terms of Equation (3) we get $nk2^{-L_1} = nk^2 2^{-2L_2}$, i.e. $2^{2L_2-L_1} = k$. Hence,

$$L_2 = \frac{L_1}{2} + \frac{\log_2 k}{2} \quad (4)$$

Equation (4) shows the relationship between L_1 and L_2 to have the same level of security. In general, we can show that if the supplier decides to send m copies of the software, then the following relationship must hold to achieve the same level of security.

$$L_m = \frac{L_1}{m} + \frac{(m-1)\log_2 k}{m} \quad (5)$$

where, L_m is the length of encryption keys when m copies of the software are to be sent. Table I shows the size of L_m needed to achieve the same level of security for different values of k (*speed ratio of hacker's computer*) and m (*number of copies of the software to be sent by the supplier*).

Table I: Size of L_m to achieve the same security level for different values of k and m.

Size of L_m (in bits) for $L_1 = 128$ bits			
k	$m=2$	$m=3$	$m=4$
2	65	44	33
8	66	45	35
32	67	46	36
128	68	48	38
512	69	49	39
2048	70	50	41

In the above analysis, it is assumed that the hacker stays with the vehicle all the time. However, it may not be possible for the hacker to stay with the vehicle 24 hours a day and 7 days a week. If the duplicate copies of the software are sent after some random and long average time intervals (say 24 to 48 hours), then the hacker may not stay near the vehicle when the vehicle will be receiving the duplicate copies of the software. Thus, in real-life, a much higher level of security can be achieved by sending two or more copies of the software to the vehicle.

D. Trade off between m and m+1 Copies of the Software

When m+1 copies of the software are sent, the length of the keys is shorter than when m copies of the software are sent. Shorter keys need less time to do encryptions and decryptions of the software. However, sending m+1 copies of the software need more time than sending m copies of the software. In our future work we would like to evaluate the trade off between m and m+1 copies of the software by considering various performance parameters such as FEMA, cost/benefit and time to deploy the upgraded software.

V. CONCLUSIONS

In this paper, we have presented architecture for secure software upload in vehicles. We proposed that the supplier should send at least two copies of the software to the vehicle. We have given a detailed description of key exchange mechanisms and software upload process. If two or more copies of the software are sent, then shorter keys can be used to obtain the same level of security that can be obtained with longer keys when only one copy of the software is sent. In other words, with the same size of keys, a significantly higher level of security can be obtained if two copies of the software are sent to the vehicle, instead of only one copy. The level of security can be further increased if the time interval between the transmission of two copies of the software is a random number with a very long average value. Since different authentication keys are used for different vehicles, neither a vehicle nor the software vendor can upload software in any other vehicles without the consent of the automotive company.

REFERENCES

[1] "Top 10 Techno-Cool Cars", IEEE Spectrum, February 2003, pp. 30 –35.

[2] A Frier, P. Karlton, and P. Kocher, "The SSL 3.0 Protocol," Netscape, Nov. 1996.

[3] M.S Bhiogade, Secure Socket layer, IS 2002 Proc. of the informing science IT Education Conf., June 19-21, 2002, Cork, Ireland, pp 0085-0090. http://ecommerce.lebow.drexel.edu/eli/2002Proceedings/papers/Bhiog058Secur.pdf

[4] How Virtual Private Networks Work. Website of how stuff works. http://computer.howstuffworks.com/vpn.htm.

[5] "Handbook of Applied Cryptography " By A. Menezes, P. Van Oorschot and S. Vandstone. CRC Press 1996.

[6] Ronald Rivest, "The MD5 Message-Digest Algorithm", RFC 1321 April 1992.

[7] Sung-Ju Lee, "Routing and Multicasting Strategies in Wireless Mobile Ad hoc Networks," Ph. D. Dissertation, Univ. of California Los Angeles, 2000.

[8] Thomas Kunz and Ed Cheng, "On-Demand Multicasting in Ad-Hoc Networks: Comparing AODV and ODMRP," Proc. of the 22nd International Conf. on Distributed Computing Sys. (ICDCS'02), IEEE Comp. Society, 2002.