1-1-2012

# Remote software upload techniques in future vehicles and their performance analysis

Irina Hossain
*Wayne State University,*

# REMOTE SOFTWARE UPLOAD TECHNIQUES IN FUTURE VEHICLES AND THEIR PERFORMANCE ANALYSIS

by

**IRINA HOSSAIN**

**DISSERTATION**

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

**DOCTOR OF PHILOSOPHY**

2012

MAJOR: COMPUTER ENGINEERING

Approved by:

Advisor Date

## DEDICATION

I dedicate this work to my family who helped and sacrificed the most during this journey.

## ACKNOWLEDGEMENTS

First, I wish to thank Almighty for all His blessings and helps.

I would like to convey my sincere gratitude and appreciation to my advisor Dr. Syed Masud Mahmud for his patience, continuous support, encouragement and guidance during the research work. His valuable feedback and constructive suggestions contributed a lot to this work. Without his support it was impossible to continue my work.

I would like to thank Dr. Cheng-Zhong Xu, Dr. Feng Lin and Dr. Pepe Siy for serving on my dissertation committee. My heartiest thank goes to Dr. Huirong Fu for being the external member in my dissertation committee.

I do not have enough words to thank Dr. Moon Ho Hwang who spent his days and nights helping me developing the simulation model for this work. I'm thankful to my brother Dr. Ekram Hossain for his valuable feedback and suggestion. Thanks to my colleagues Dr. Zaydoun Rawashdeh, Dr. Nader Mazen Rabadi and Dr. Radovan Miucic. Any time I needed any kind of help, they were always there.

Lastly, I would like to remember my husband and each of my family members who provided me continuous support and inspiration. Thanks to my parents for everything they did for me to achieve all the accomplishments in my life.

## TABLE OF CONTENTS

**LIST OF FIGURES**

## LIST OF TABLES

# 1 CHAPTER 1 - INTRODUCTION

## 1.1 Motivation

Today's vehicles have hundreds of microcontroller based Electronic Control Units (ECUs) such as Engine Control Unit, Transmission Control Unit, Antilock Brake System, Adaptive Cruise Control Unit, Air-bag Control Unit and so on. Moreover, the Intelligent Transportation System (ITS) has taken initiatives to use wireless communication technologies such as Vehicle-to-Roadside (V2R), Vehicle-to-Vehicle (V2V) and Vehicle-to-Person (V2P) communications to improve road safety, transportation efficiency, traveling comfort, and provide ubiquitous wireless connectivity to the Internet. Various mobile commerce services such as navigation information, emergency roadside assistance, location-based services, deliverance of digital information such as e-mail, entertainment, diagnostics and prognostics, pay for insurance can be incorporated in future vehicles [1]. Integration of these new features will require future vehicles to host more ECUs, communication devices and other hardware and software modules. The software of these ECUs needs to be updated from time to time for a variety of reasons such as to add new applications, add/upgrade new functionalities in existing system, tune performance parameters, eliminate software bugs, avoid recalls and keep the vehicle compatible with the ITS infrastructure. Thus, software update in vehicles' ECUs will become a routine task for the manufacturers in near future.

Present software update technique in vehicles' ECUs is a unicast process and it is done physically, e.g., software is uploaded on an individual basis in service station.

Instead of updating in service station if the software could be updated remotely using wireless communication links, it would be beneficial to both users and auto manufacturers in terms of time, labor and money. In this technique, an ECU will receive the software from a remote software distribution center using the road side Base Station (BS) as a gateway between the vehicle and the remote download server. However, as with any other wireless applications, security is a crucial issue in remote software updates using wireless link. Transmitting software packets over radio channels makes eavesdropping, data altering, theft of service, and denial of service (DoS) attacks easier for adversaries. The basic security requirements, i.e., **integrity** and **confidentiality** of the transmitted software and **authenticity** of both the software sender and receiver must be guaranteed in order to avoid any future disasters due to malfunctions of the vehicle or to protect the proprietary algorithms from hackers, competitors or people with malicious intent. Since security protocols in WLAN and WWAN has security flaws [2, 3], additional security mechanism is needed for RSU. This thesis presents a secure architecture for RSU in a vehicle's ECU where the software provider can establish a point-to-point secure communication link with the vehicle via the roadside BS under which the vehicle resides and sends the necessary software to the non-functioning ECU.

The above software upload technique is useful if a particular vehicle experiences some problems with its functionality. However, if an Auto Company or ITS authority decides to add new features or update an existing features or needs to eliminate software bug from a large number of vehicles, multicasting the software packets will be more efficient than multiple unicast transmission. Multicasting is the bandwidth conserving technology that accepts a single message from an application and delivers copies of the message to a subset of hosts as a group transmission [4]. As with unicast, multicast communication requires security services such as access control, data confidentiality, traffic confidentiality, integrity, data authentication, source authentication, sender and receiver non-repudiation and service assurance. In this thesis, an infrastructure based secure wireless multicasting architecture is presented for RSU in future vehicles' electronic modules. In general, the security requirements are achieved by employing modern cryptographic techniques using a session key (SK) shared by all legitimate group members. The design of an efficient key management technique i.e., generation, distribution and updating the SK is the critical part for the realization of secure multicast. Since distribution and re-keying of SK incur extra burden in communication network, key management technique seeks to minimize this computation and communication burden by efficient key distribution as well as providing security requirements. Hence, one of the objectives of this research is to determine an efficient and reliable key management protocol for Vehicular Software Distribution Network (VSDN). The VSDN is considered as wireless network where vehicles are connected to the software distributors through roadside BSs. In order to make the system scalable the large VSDN is partitioned into small subgroups based on region and the Regional Group Manager (RGM) will be responsible for distributing the software within its region. The Software Vendor (SV) named as Central Manager (CM), RGMs, BSs and vehicles form hierarchical structure where the CM sends the software packets to RGMs and the RGM transmits the packets to BSs within that region and finally, the BSs multicasts these to authorized vehicles. This hierarchical architecture benefits from its decentralized nature of Group Key Management (GKM) [5]. Although the BSs are not interested in multicast data, they can take active part in key management to reduce the workload on multicast Group Manager (GM). This requires some level of trust on the BSs [6]. On the other hand, one of the security requirements of an efficient GKM protocol is to put trust on minimal number of entities to increase the security level [7]. In this research, two different scenarios are considered. In one scenario the BSs are considered as fully-trusted and in other case they are considered as semi-trusted entities. Consequently, two GKM protocols are proposed and a comparative performance analysis of these two protocols is presented in terms of computation overhead, communication overhead and storage overhead required by each entity involved in key management. Finally, how the overhead on GM and vehicles can be traded-off with the level of trust on BSs is discussed.

## 1.2 Contribution

The contribution of the thesis can be summarized below:

1. This work presents a mechanism for RSU in a vehicle's ECUs when that particular vehicle encounters some problem with its functionality. In this mechanism, the AC or SV sends software to the vehicle via roadside BS. The BS authenticates the vehicle using vehicle's authentication key which is pre-loaded to the vehicle and establishes secure point-to-point wireless communication link before sending the software to the vehicle. In order to increase the security level, it is recommended to send multiple copies of the

software. Through analytical and simulation results it is shown that it is sufficient to send two copies of software to the vehicle.

2. In order to upload software to a large number of vehicles at a time, this work also proposes a secure wireless multicast protocol that employs BSs as gateways between vehicle and software provider.
3. Consequently, two GKM protocols for secure multicast are proposed based on the trust level on BSs. Both protocols provide mutual authentication, non-repudiation and secure transfer of multicast SK to the group members.
4. The performance of the proposed GKM protocols is compared in terms of communication, computation and storage overhead. Analytical model is developed to obtain the communication delay while transmitting messages through wired and wireless channel. The models are used to find various latency such as multicast session initialization latency, handoff latency etc.
5. Extensive simulation is conducted to validate the analytical results of multicast session initialization and find the feasibility of the proposed models.

In future, vehicles will be equipped with wireless devices to communicate with each other as well as with various service providers. The AC can use wireless link to update software to various ECUs remotely. This research will provide solution on how the software packets could be distributed efficiently, securely and cost effectively.

## 1.3 Outline

Subsequent chapters of this thesis are organized as follows:

**Chapter 1 –** This chapter provides an overview of the research work that will be done throughout the dissertation. It describes the motivation behind this work and defines the research objectives.

**Chapter 2 –** This chapter presents background information about the technology used in this work. It also gives a brief review of existing security protocols used in wireless communication and past research done on multicast key management. Each technology and past work is discussed with a concise expository of its technique and performance evaluation if it is available.

**Chapter 3 –** This chapter describes the secure unicast software upload technique and its analytical modeling for different scenarios with regard to vehicle's buffer size and software packet verification methods.

**Chapter 4 –** This chapter aims to describe the proposed multicast architecture to upload software in future vehicles' electronic modules using wireless technology. It explains the detail key management technique used for secure multicast. It also develops analytical model of the whole system to find the end-to-end delay on establishing the secure group communication and to find the handoff latency during on-going multicast session.

**Chapter 5 –** This chapter summarizes the results of performance analysis of both the unicast RSU technique and multicast RSU technique and provides discussions on the results.

**Chapter 6 –** This chapter concludes the thesis.

# 2 CHAPTER 2 – BACKGROUND AND RELATED WORKS

This chapter gives a brief overview of basic technology used in this thesis, current security protocols exist in wireless communication technology, various GKM protocols proposed in past and the past research done on RSU.

## 2.1 MAC Protocol for IEEE 802.3

The IEEE has developed the 802.3 standard families for local and metropolitan area networks. The 802.3 MAC protocol employs 1-persistent Carrier Sense Multiple Access protocol with Collision Detection (CSMA/CD) [8]. The operation of 1-persistent CSMA/CD transmitter and receiver are shown in Figure 2.1 and Figure 2.2, respectively. If the channel is sensed idle, it transmits the packet with probability 1. If the channel is sensed busy, it waits until the channel goes idle and transmits the packet. With the collision detection feature, a station can "listen while transmit". With this feature all the data frame transmission involved in a collision will be aborted immediately after the collision is detected, thus reduces the duration of a collision period. The collided stations broadcast jam signal to make the collision more obvious. The retransmissions of the unsuccessful transmissions are scheduled at some time later in future. In case of collision, the truncated Binary Exponential Backoff (BEB) [9] retransmission algorithm is used to resolve the collision.

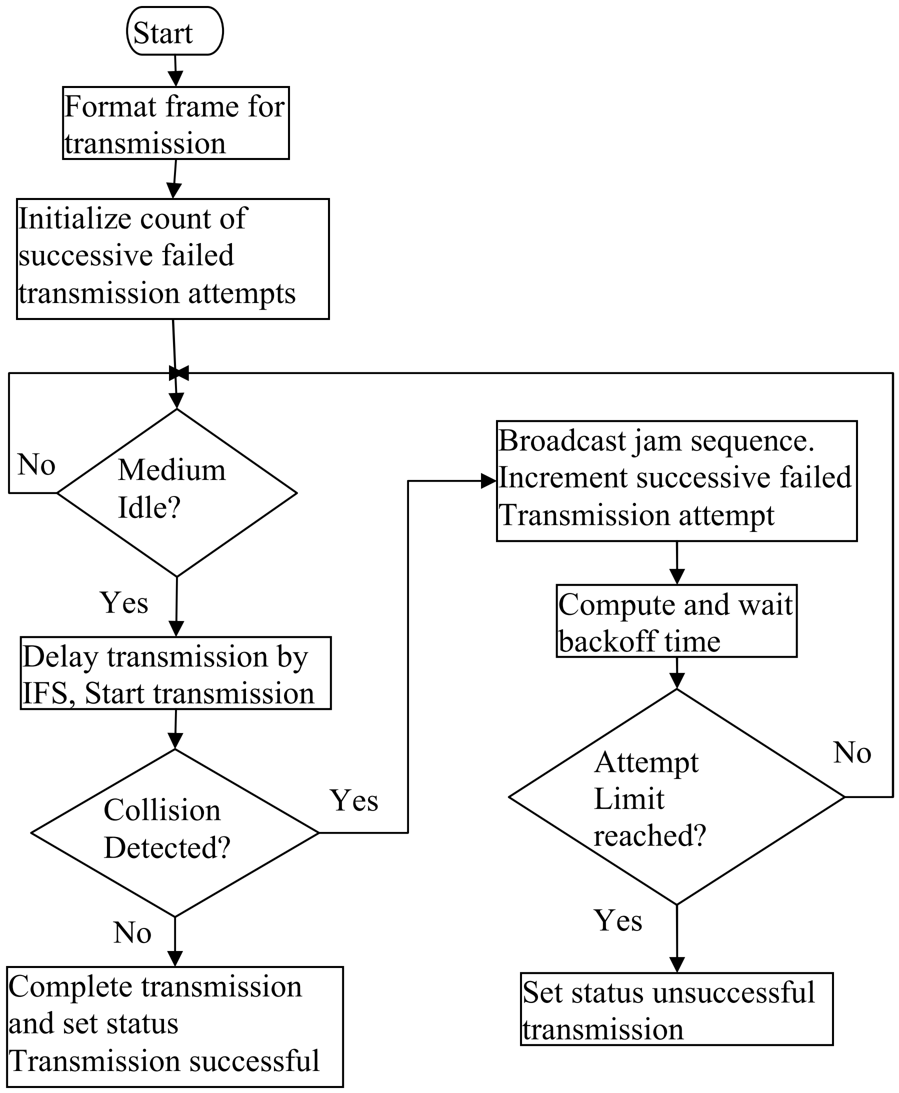
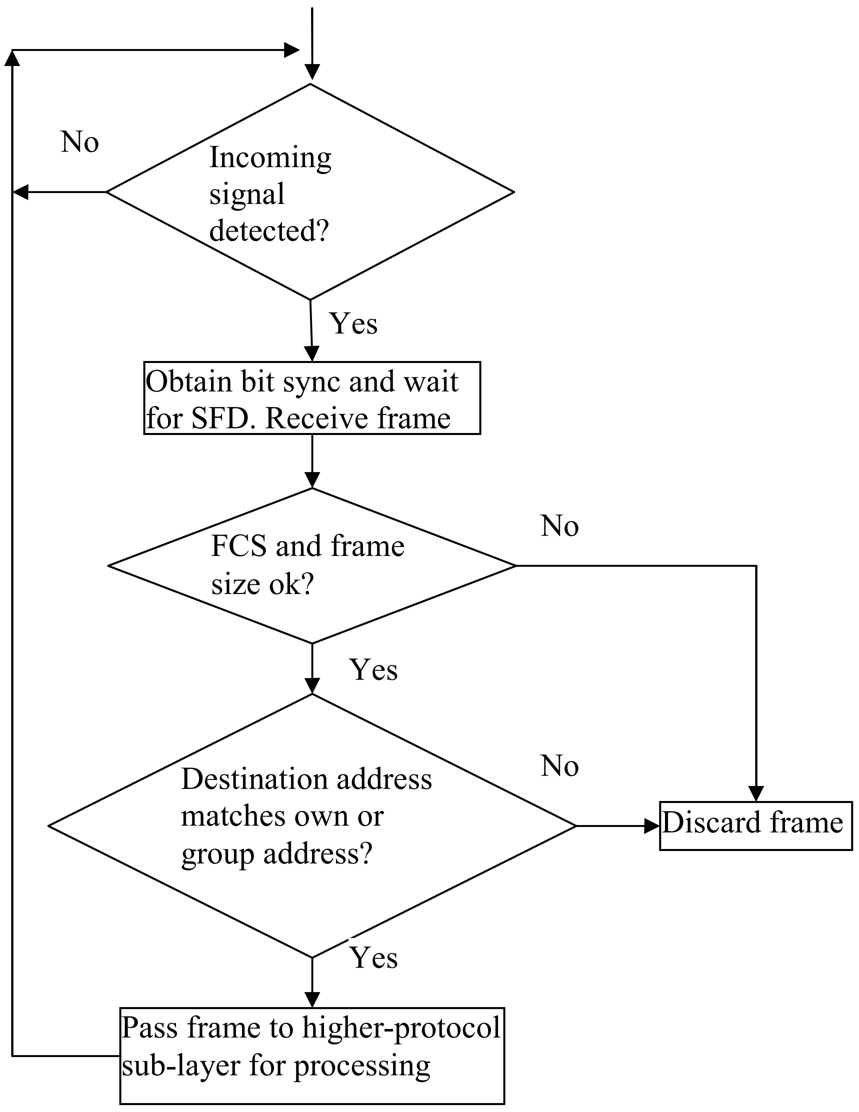

Start
Format frame for transmission
Initialize count of successive failed transmission attempts
No
Medium Idle?
Yes
Delay transmission by IFS, Start transmission
Collision Detected?
Yes
No
Complete transmission and set status Transmission successful
Broadcast jam sequence. Increment successive failed Transmission attempt
Compute and wait backoff time
Attempt Limit reached?
No
Yes
Set status unsuccessful transmission

**Figure 2.1 1-persistent CSMA/CD Transmitter.**

### 2.1.1 Delay Analysis of 1-persistent CSMA/CD

Several throughput and delay analysis for 1-persistant CSMA/CD are available in literature [10-13]. In this work, the analysis of 1-persistent CSMA/CD protocol performed by Iida *et. al.* [12] and Tobagi *et. al.* [10] and BEB retransmission algorithm is used to find the average message transmission delay for wired part of the


No
Incoming signal detected?
Yes
Obtain bit sync and wait for SFD. Receive frame
FCS and frame size ok?
No
Yes
Destination address matches own or group address?
No
Discard frame
Yes
Pass frame to higher-protocol sub-layer for processing

network.

**Figure 2.2 1-persistent CSMA/CD Receiver.**

According to [12], a transmission cycle of a channel consists of a busy period (*B*) and an idle period (*I*). Let *U* denotes the useful period in a cycle where the channel carries useful information. Using renewal theory the average channel utilization,

$$S = \frac{E(U)}{E(B+I)} = \frac{E(U)}{E(B)+E(I)} = \frac{\overline{U}}{\overline{B}+\overline{I}} \qquad \textbf{(2.1)}$$

In a busy period there could be three different types of transmission period (TP) depending on success and collision:

$$T(1) = 1 + a$$
$$T(2) = Y + 2a + b \qquad \textbf{(2.2)}$$
$$T(3) = 2a + b$$

where $T$ is the packet transmission time, $a = \frac{\tau}{T}$ is the normalized propagation delay ; $b = \frac{J}{T}$ is the normalized jamming signal and $Y$ is the time interval between the beginning of TP and the generation time of the first colliding packet within the vulnerable period $a$. The cumulative distribution function of $Y$ is:

$$F_Y(y) = \frac{e^{-Gy} - e^{-aG}}{1 - e^{-aG}} \qquad \textbf{(2.3)}$$

where $G$ is the offered load i.e., the average number of packets (both new and retransmitted ) generated per packet duration time.

Let, $P^i(j)$ is the probability that $TP_i$ , the $i^{\text{th}}$ TP, is equal to $T(j)$ and $r_i$ , the probability that there is no transmission queue at the end of $TP_i$. The expected values of $B$ , $U$ , and $I$ can be calculated as:

$$\overline{B} = \sum_{k=1}^{\infty} \rho_k \sum_{i=1}^{k} \sum_{j=1}^{3} P^i(j) \overline{T(j)}$$
$$\overline{U} = \sum_{k=1}^{\infty} \rho_k \sum_{i=1}^{k} P^i(1) \qquad \textbf{(2.4)}$$
$$\overline{I} = \frac{1}{G}$$

where $\rho_k$ is the probability that a busy period (*BP*) consists of *k TP*s. The average collision time is given by,

$$\overline{C} = \overline{T(2)} = \overline{Y} + 2a + b$$
$$= 2a + b + \frac{1}{G} - \frac{ae^{-aG}}{1-e^{-aG}} \qquad \textbf{(2.5)}$$

Unlike other access method, 1-persistent CSMA channel incurs an additional pre-transmission delay, $W$ , if upon its arrival, the packet detects the channel busy. The average waiting time when the channel is sensed busy is given by $\frac{\overline{Z}^2}{2\overline{Z}}$ where $Z$ is the random variable representing the time $B'$ during a cycle that the channel is in its busy period excluding the first $a$ seconds. The average value of $Z$ is given by

$$\overline{Z} = \sum_{k=1}^{\infty}\rho_k\sum_{i=1}^{k}\frac{1}{i}\sum_{j=1}^{3}P^i(j)(\overline{T(j)}-a)$$
$$\overline{Z^2} = \sum_{k=1}^{\infty}\rho_k\sum_{i=1}^{k}\frac{1}{i}\sum_{j=1}^{3}P^i(j)\overline{(T(j)-a)^2} \qquad \textbf{(2.6)}$$
$$\overline{B'} = \sum_{k=1}^{\infty}\rho_k\sum_{i=1}^{k}\sum_{j=1}^{3}P^i(j)\overline{T(j)}-a$$

Then the average waiting time, $W$ is given by

$$\overline{W} = \frac{\overline{B'}}{\overline{B}+\overline{I}}\frac{\overline{Z^2}}{2\overline{Z}} \qquad \textbf{(2.7)}$$

The average number of retransmission for successful transmission is given by [10]

$$N_r = \frac{G}{S} - 1 \qquad \textbf{(2.8)}$$

For *BEB* algorithm, the average backoff time for $n^{th}$ retransmission [14],

$$t_{backoff_n} = \begin{cases} t_b \dfrac{K-1}{2}, & n \le m \\ t_b \dfrac{M-1}{2}, & n > m \end{cases} \quad \text{where } K = 2^n, \quad M = 2^m \tag{2.9}$$

where $t_b$ is the 1-unit of backoff time, *m* is the maximum backoff window size.

The probability of collision at $n^{\text{th}}$ retransmission [14]

$$P_{c_n} = \frac{1}{2^n} \tag{2.10}$$

Hence, average backoff time for $N_r$ number of retransmissions,

$$\overline{t_{backoff}} = \sum_{n=1}^{N_r} P_{c_n} t_{backoff_n} = t_b \sum_{n=1}^{N_r} \frac{1}{2^n} t_{backoff_n} \tag{2.11}$$

Let $R$ be the average delay between two consecutive transmissions (i.e., a retransmission) of a given packet. For 1-persistent CSMA/CD,

$$R = \overline{C} + \overline{t_{backoff}} + \overline{W} \tag{2.12}$$

Finally, the expected packet delay is

$$t_D = (\frac{G}{S} - 1)R + 1 + a + \overline{W} \tag{2.13}$$

## 2.2 MAC Protocol for IEEE 802.11

The IEEE 802.11 working group has decided the CSMA with Collision Avoidance (CSMA/CA) as MAC protocol for wireless LANs using BEB algorithm for medium access during collision [15]. The fundamental technique is called Distributed Coordination Function (DCF). The DCF defines two access mechanisms: the basic access and the Request-to-Send/Clear-to-Send (RTS/CTS) mechanisms. In

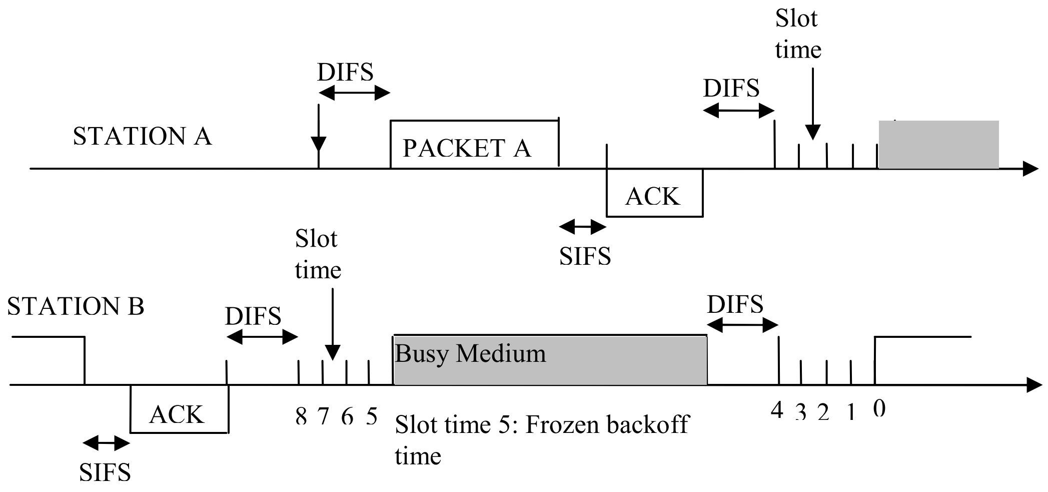


**Figure 2.3 Example of Basic Access Mechanism.**

this thesis, basic access mechanism is assumed since the GKM messages are short. An example of data transmission using basic access method is shown in Figure 2.3. In the basic access method, a station with a packet to transmit senses the medium. The station transmits if the medium is idle. If the medium is sensed busy, it continues to monitor the medium until it becomes idle for more than Distributed Inter-Frame Space (DIFS) time period. The station then initializes the backoff timer and defers transmission for a randomly selected backoff interval in order to minimize the collision. The backoff timer is decremented when the medium is idle, is frozen when the medium is sensed busy and resumes only after the medium has been idle for longer than DIFS. The station, whose backoff timer expires first, begins transmission and the other stations freeze their timers and defer transmission. Once the current station completes transmission, the backoff process repeats again and the remaining stations re-activate their backoff timer. A station that receives a data packet, replies with a positive acknowledgement (ACK) packet after a SIFS time interval, confirming the successful reception of the packet. If the sender does not receive an ACK within specified time, the packet is assumed to be lost and reschedule according to the backoff rules. In addition, a station must wait a random backoff time between two consecutive packet transmissions to avoid collision. After a successful packet transmission, if the station has packets buffered for transmission, it starts a new backoff process.

### 2.2.1 Delay Analysis of CSMA/CA

Researchers have used two-dimensional Markov chain model of its backoff window size to compute throughput and average transmission delay of IEEE 802.11 MAC protocol [16-18]. Following the same reasoning with [16-18], the probability $\tau$ that a station transmits a packet in a randomly chosen slot time is given by

$$\tau = \begin{cases} \dfrac{2(1-p^{m+1})}{1-p^{m+1}+(1-p)W\left(\sum_{i=0}^{m}(2p)^i\right)}, & 0 \le i \le m' \\ \dfrac{2(1-p^{m+1})}{1-p^{m+1}+pW\left(\sum_{i=0}^{m'-1}(2p)^i\right)+W(1-2^{m'}p^{m+1})}, & m' \le i \le m \end{cases} \qquad \textbf{(2.14)}$$

where $W$ is the initial window size, $m$ is the station retry limit, $m'$ is the number of backoff stages and $p$ is the probability of an unsuccessful (re)transmission seen by a station while transmitting a packet through the channel. The unsuccessful (re)transmission can happen due to: collision with at least one of the $(n-1)$ remaining stations, occurring with probability $p_c$ , or packet error due to fading and/or noise. Let $p_f$ is the probability of packet error due to fading/noise. Then the pseudo collision probability $p$ can be expressed as:

$$
\begin{aligned}
p &= p_c + p_f - p_c p_f \\
&= 1-(1-\tau)^{n-1} + p_f - (1-(1-\tau)^{n-1})p_f
\end{aligned} \qquad \textbf{(2.15)}
$$

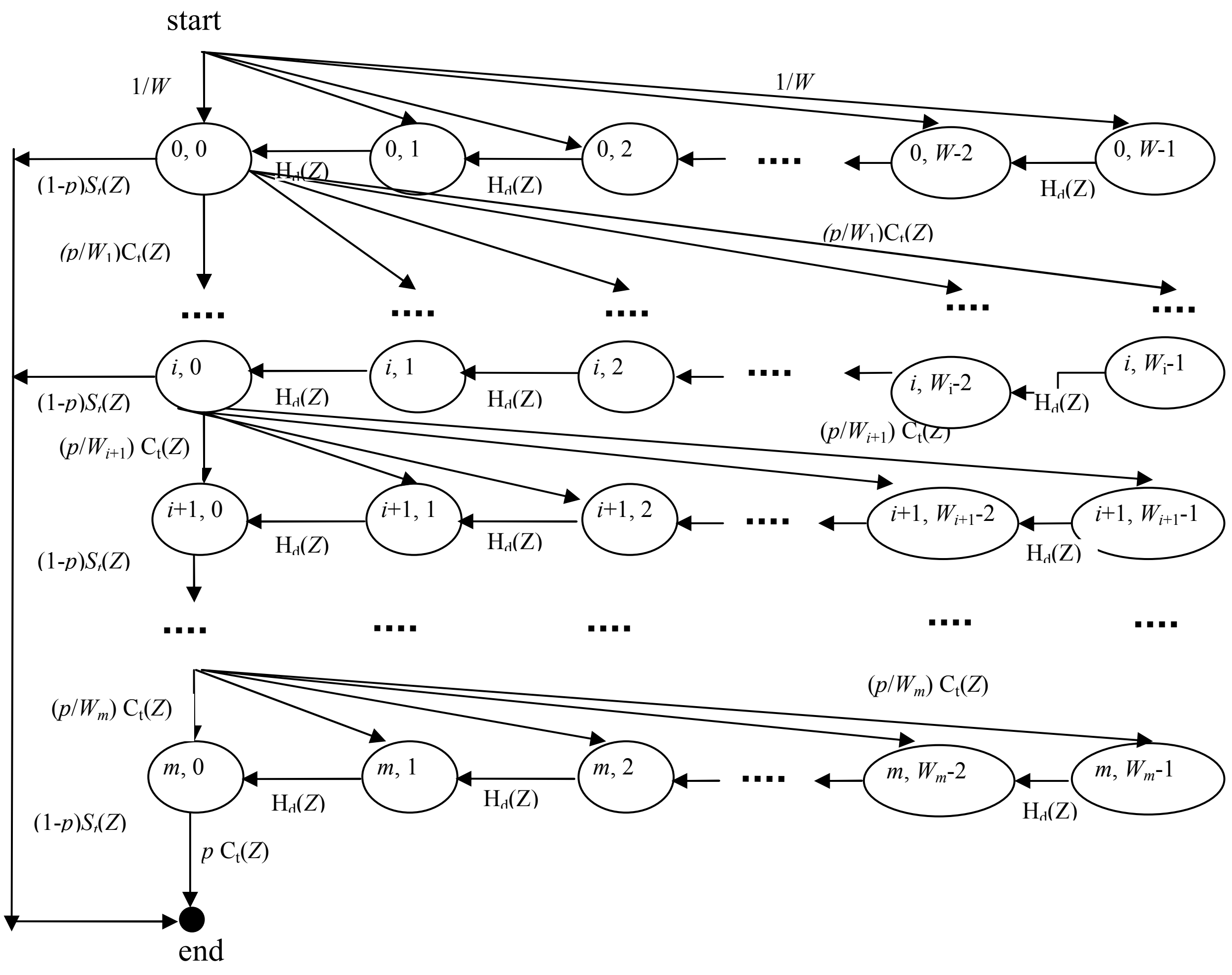


**Figure 2.4 Generalized state transition diagram of transmission process based on Markov Chain of backoff window size.**

Equations (2.14) and (2.15) represent a non-linear system with two unknowns $\tau$ and $p$ which can be solved numerically.

**Figure 2.5 Time for successful packet transmission and collision of Basic Access Method.**

In [19], the Markov Chain model of the exponential backoff process and the signal transfer function of generalized state transition diagram is used to find the probability distribution of MAC layer service time. As mentioned earlier, the backoff timer will be decremented by one for every idle slot detected, will be suspended and deferred a time period of $T_{suc}$ or $T_{col}$ when the medium is detected as busy due to a successful transmission or a collision, respectively. The signal transfer function, $H_d(Z)$ of the decrement process of backoff timer is given by [19],

$$H_d(Z) = \frac{(1-p)Z^{\sigma}}{[1-P_{suc}S_t(Z)-(p-P_{suc})C_t(Z)]} \qquad \textbf{(2.16)}$$

where $\sigma$ is the duration of an empty slot, $S_t(Z)$ and $C_t(Z)$ are probability generating function (PGF) of $T_{suc}$ and $T_{col}$, respectively and $P_{suc}$ is the probability that there is one successful transmission among other stations in the considered time slot given that the current station does not transmit. For basic access method, $T_{suc}$ and $T_{col}$ is (Figure 2.5)

$$T_{suc} = H + t_p + SIFS + ACK + DIFS + 2\delta$$

$$T_{col} = H + t_p + DIFS + \delta \qquad \textbf{(2.17)}$$

where, $H$ is the MAC and PHY header, $t_p$ is the packet transmission time and $\delta$ is the propagation delay. Assuming fixed packet size, the PGF of $T_{suc}$ and $T_{col}$ is given by

$$S_t(Z) = Z^{H+t_p+SIFS+ACK+DIFS+2\delta} = Z^{\alpha}$$

$$C_t(Z) = Z^{H+t_p+DIFS+\delta} = Z^{\beta} \qquad \textbf{(2.18)}$$

Let $T_D$ is the time elapsed from the start state (i.e., beginning to be served) and end state (i.e., being transmitted successfully or discarded after maximum times retransmission failures) of a packet transmission process. According to [19], the PGF $B(Z)$ of $T_D$ is given by,

$$HW_i(Z) = \begin{cases} \sum_{j=0}^{2^i W-1} \frac{H_d^j(Z)}{2^i W}, & 0 \le i \le m' \\ HW_{m'}(Z) & m' < i \le m \end{cases} \tag{2.19}$$

$$H_i(Z) = \prod_{j=0}^{i} HW_j(Z), \qquad 0 \le i \le m \tag{2.20}$$

$$\begin{aligned} B(Z) &= (1-p)S_t(Z)\sum_{i=0}^{m}\left(pC_t(Z)\right)^i H_i(Z) + \left(pC_t(Z)\right)^{m+1} H_m(Z) \\ &= (1-p)Z^{\alpha}\sum_{i=0}^{m}(pZ^{\beta})^i H_i(Z) + (pZ^{\beta})^{m+1} H_m(Z) \end{aligned} \tag{2.21}$$

The $B(Z)$ can be expanded in power series as:

$$B(Z) = \sum_{i=0}^{\infty} \Pr(T_D = i) Z^i \tag{2.22}$$

Hence, the arbitrary $n^{th}$ moment of $T_D$ can be obtained from the $n^{th}$ differentiation of $B(Z)$. For example, the mean and second moment of $T_D$ is given by:

$$t_D = E(T_D) = \frac{dB(Z)}{dZ}\Big|_{Z=1}$$

$$E[T_D^2] = (z\frac{d}{dZ})^2 B(Z)\,|_{Z=1} = \left\{ z\frac{d}{dZ} * z\frac{dB(Z)}{dZ} \right\}|_{Z=1} \tag{2.23}$$

## 2.3 Authenticated Diffie-Hellman Key Exchange Method

The Diffie-Hellman (D-H) key exchange is a well-known cryptographic protocol that allows two users to jointly establish a shared secret over an insecure communication channel without any prior secrets [20]. The protocol uses two system parameters: $p$ is a prime number and $g$ (usually called generator) is an integer less than $p$ which is capable of generating every element from 1 to $p-1$ when multiplied by itself a certain number of times, modulo the prime $p$. The algorithm is described below.

Suppose Alice and Bob want to agree on a shared secret key using D-H key agreement protocol.

1. Alice generates a random private value $a$, computes public value $g^a \bmod p$ and sends it to Bob.
2. Bob generates a random private value $b$, computes public value $g^b \bmod p$ and sends it to Alice.
3. Alice computes $k_{ab} = (g^b \bmod p)^a \bmod p$ and Bob computes $k_{ba} = (g^a \bmod p)^b \bmod p$.
4. Now both Alice and Bob have the shared key $k = k_{ab} = k_{ba}$

The protocol depends on the discrete logarithm problem for its security. It assumes that it is computationally infeasible to calculate the shared secret key $k$ given the two public values $g^a \bmod p$ and $g^b \bmod p$ when the prime $p$ is sufficiently large.

Since the protocol does not provide authentication of the communicating parties, the D-H key exchange is vulnerable to man-in-the-middle attack. A person in the middle may establish two distinct D-H key exchanges, one with Alice and one with Bob, effectively masquerading Alice to Bob and vice versa. In 1992, the authenticated D-H key agreement protocol was developed to overcome the man-in-the-middle attack [21]. The security is achieved by allowing two parties to authenticate themselves to each other by using public-key certificate and digital signatures. It assumes that the two parties Alice and Bob possess a public/private key pair and a certificate for the public key. Both Alice and Bob sends their signature and public-key certificate along with their public values $g^a \bmod p$ and $g^b \bmod p$ , respectively. Even the attacker can intercepts messages between Alice and Bob, she cannot forge signatures without their private keys.

## 2.4 WLAN Security for 802.11

Although wireless technologies offer the users with additional conveniences over the wired technologies, they also introduce unique security challenges. Various threats and vulnerabilities associated with wireless network and hand held devices are listed in [2, 3]. Broadcasting messages over radio channels makes eavesdropping, data altering, theft of service, and denial of service (DoS) attacks easier for adversaries. Hence, additional mechanisms are needed to protect the communication over wireless network. Brief descriptions of current security protocols used in WWAN (3G cellular) and WLAN (802.11) are shown below. Since WPAN allows very short distance communication, it is not suitable for remote software upload. Hence, security of WPAN is not discussed here.

### 2.4.1 WTLS, KSSL, WEP, WPA and 802.11i, 802.16

**Wireless Transport Layer Security (WTLS)**

Wireless Transport Layer Security (WTLS) is the security layer of Wireless Application Protocol (WAP). The WAP is developed by WAP forum [22] to enable small, low-power mobile devices such as PDAs, cell phones to access Internet over the low bandwidth wireless networks. Due to protocol incompatibilities, WAP uses

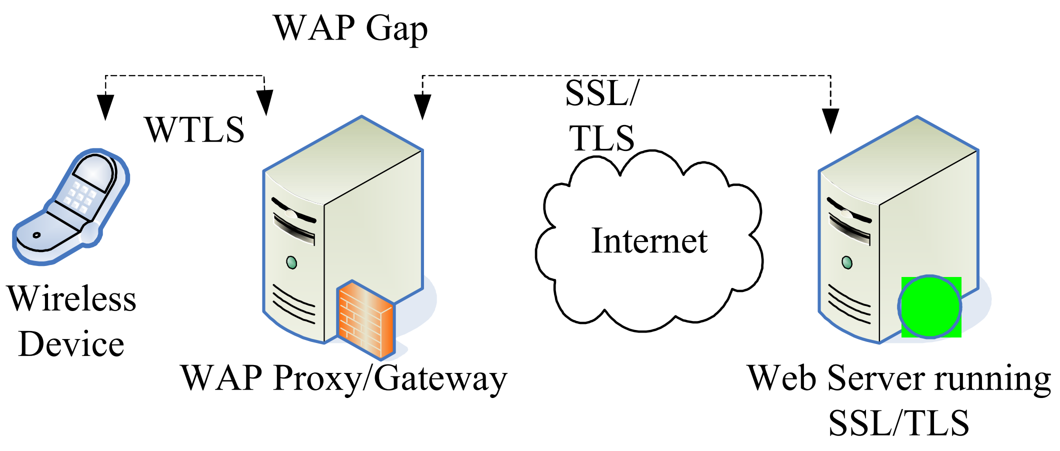


**Figure 2.6 WTLS: A Proxy Based Protocol.**

proxy-based technology to connect wireless domain to Internet as shown in Figure 2.6. The WAP proxy server or gateway translates requests from WAP protocol to Internet Protocol (takes place between Web server and WAP proxy/Gateway) and translates WAP content into compact encoded format to send over low-bandwidth wireless network (takes place between wireless client and WAP proxy/Gateway) [23, 24]. The WAP required two security protocols: WTLS from wireless client to the WAP Gateway and SSL/TLS from the WAP Gateway to a web server on the Internet as shown in Figure 2.6. During the protocol translation from WTLS to TLS, WTLS encrypts the communication between the client and gateway, decrypts and re-encrypts using SSL/TLS, leaves data temporarily unencrypted in WAP Gateway. This so called "WAP gap" problem and many other security problems identified in [25, 26] demanded additional improvement of WAP protocol. The WAP 2.0 includes a version of TLS in WAP device stack so that WAP Gateway no longer needs to translate WTLS protocol into TLS protocol, thus provides better end-to-end security [24].

**Kilobyte SSL (KSSL)**

Although SSL/TLS is the most widely used security protocol for secure transaction over the wired Internet, it was assumed to be inadequate for resource-constrained devices. Vipal Gupta and Sumit Gupta from Sun Microsystems investigated the performance of SSL on mobile devices and found that SSL is also suitable for wireless devices [27]. They developed Kilobyte SSL (KSSL) which is a small-footprint, client-side-only implementation of SSL 3.0 for hand held and wireless devices, supports RSA_RC4_128_MD5 and RSA_RC4_40_MD5 cipher suites, provides server authentication via RSA signatures as well as arbitrary certificate chain lengths and uses "abbreviated SSL handshake" SSL architecture. The result was corroborated by [28] where the feasibility of using strong cryptographic protocols such as SSL, S/MIME and IPSec on mobile devices was investigated. The increase of processing power and memory, and the feasibility of using SSL/TLS in hand-held devices has motivated researchers to design SSL/TLS based Authentication and Key Agreement (AKA) for future mobile networks [29, 30].

**Wired Equivalent Privacy (WEP)**

Wired Equivalent Privacy (WEP) is the first protocol which attempted to provide security in IEEE 802.11 standards [15]. WEP provides encrypted communication between a wireless access point (AP) and a client. The communicating parties share a 40-bit (or 104-bit) secret key k which is used to encrypt the transmitting data. The protocol is described briefly in Figure 2.7.

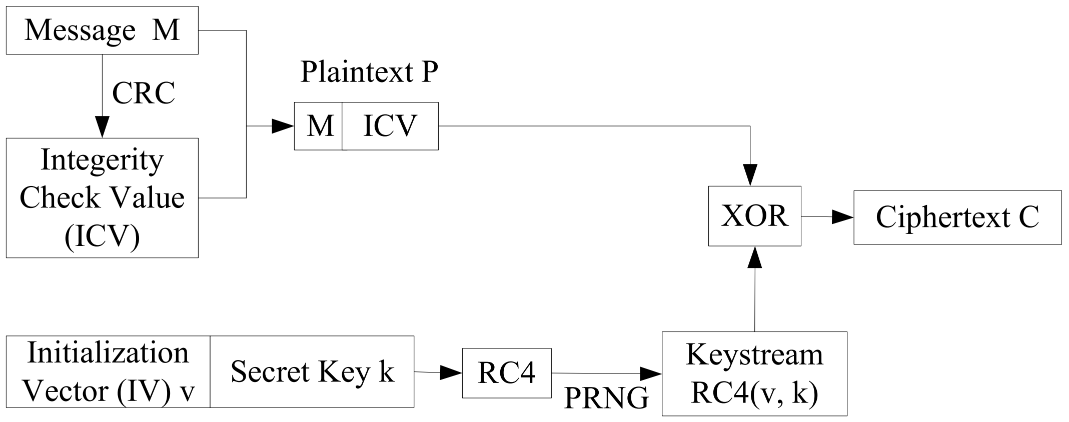


**Figure 2.7 WEP Protocol.**

IEEE 802.11b defines other mechanisms to limit the access to an AP or a set of APs: SSID (Service Set Identifier) and MAC (Media Access Control) Address Filtering. Each AP is programmed with an SSID which corresponds to a specified wireless network [24]. A wireless station willing to access that particular AP, must be configured with the appropriate SSID and must present the correct SSID to access the AP. In MAC Address Filtering technique, each AP is configured with a list of MAC addresses associated with the client computers allowed to access the AP. A client is allowed to access the AP only if it's MAC address is enlisted and the SSID presented by the client is the AP's SSID.

Unfortunately, in the last few years the researchers have proved that the WEP is failed to provide three main security goals, i.e., confidentiality, integrity and authentication, due to its various design flaws and poor implementations [31, 32]. Some of them are listed below:

- 40-bit shared key is too small for brute-force attack [32]. Even though WEP allows 104-bit key, it is impossible to avoid reuse of keystream because of small Initialization Vector (IV) size (24-bit) and the static shared key. The XOR operation of two ciphertext using same IV eliminates the keystream and results in the XOR of two plaintexts. If one of the plaintexts is known, the other can be obtained. A dictionary can then be created that specifies the keystream used for each IV. In this way, an attacker can eventually decrypt all transmissions without ever knowing the secret key.
- The inherent weaknesses in the key scheduling algorithm of RC4 [33], allow an adversary to discover the key by eavesdropping several millions packets [34].
- The Integrity Check Value (ICV), used to provide message integrity, is obtained from CRC checksum operation. CRC is not a cryptographic function, rather it is an error detection function used in communication. Moreover, ICV is a linear and un-keyed function of message [32]. Hence, the integrity of message is not guaranteed. The weak integrity also causes easier plaintext recovery, IP re-direction, reaction attack etc. [32].
- Since IV reuse is allowed, message injection is possible if the attacker knows the plaintext/ciphertext pair. The keystream can be derived as Plaintext XOR Ciphertext and any plaintext can be encrypted using the keystream [32].
- Similarly, an adversary can easily spoof the Shared Authentication by eavesdropping authentication process of legitimate clients, i.e., by observing

plaintext challenge sent by the AP and encrypted response sent by the client [31, 32].

- The 802.11 standard does not specify the key management of the shared secret key, i.e., how to distribute the keys among the legitimate users [32]. Moreover, use of same key by many users makes it difficult to replace the compromised key materials [15].

**Wi-Fi Protected Access (WPA)**

The severe security flaws discovered in WEP have led the Wi-Fi Alliance to come up with strong, interoperable security replacement for WEP which is known as Wi-Fi Protected Access (WPA) [35]. WAP adopts Temporal Key Integrity Protocol (TKIP) to provide strong data encryption which still use RC4 algorithm, but includes a per-packet key mixing function, a Message Integrity Check (MIC) named Michael, an extended IV with sequencing rules and re-keying mechanism [35]. WPA provides Pre-Shared-Key (PSK) authentication for home or office environment to authenticate the peers whereas it uses IEEE 802.1X authentication [36] with an Extensible Authentication Protocol (EAP) [37] in enterprise environment to provide stronger authentication. Although WPA was proposed to address all known vulnerabilities of WEP, there are still some weaknesses due to the limitation of using previously designed hardware. The IEEE Standard Board has approved 802.11i on June 24, 2004, which incorporates more comprehensive solution for WLAN security [38]. 802.11i defines Robust Security Network Association (RSNA) for mutual authentication and key management, enhanced encryption technique called Counter-mode/CBC-MAC (cipher block chaining (CBC) with message authentication code (MAC)) to provide strong authentication, confidentiality, integrity and replay protection. The Wi-Fi Alliance has adopted the 802.11 security standard in their 2nd version of WPA known as WPA2, which was designed to meet 802.11i security criteria. The security analysis of 802.11i shows that it provides satisfactory authentication, data confidentiality, integrity, and replay protection [39]. However, several vulnerabilities such as Man-in-the-Middle attack, Security Level Rollback attack etc. might arise due to poor implementation of RSNA [39]. Moreover, the DoS vulnerabilities are not taken care of in 802.11i. Hence, it is necessary to deploy security mechanisms to defend against DoS attack to make 802.11i more robust against security threats.

## 2.5 Multicasting

Multicasting is the ability of a communication network to accept a single message from an application and to deliver copies of the message to multiple selected recipients at different locations [40]. It is a more efficient technique for group communication as it provides transmission and routing of packets to multiple destinations at a lower network and host overhead than broadcasting to all hosts or unicasting to each host in a group. Unicast (one receiver) and Broadcast (all receivers) are two special cases of the Multicast method. Figure 2.8 illustrates the difference between unicast, broadcast and multicast. Multicasting is implemented efficiently on broadcast LAN such as Ethernet. Example of such multicast is Internet Protocol (IP) multicasting.

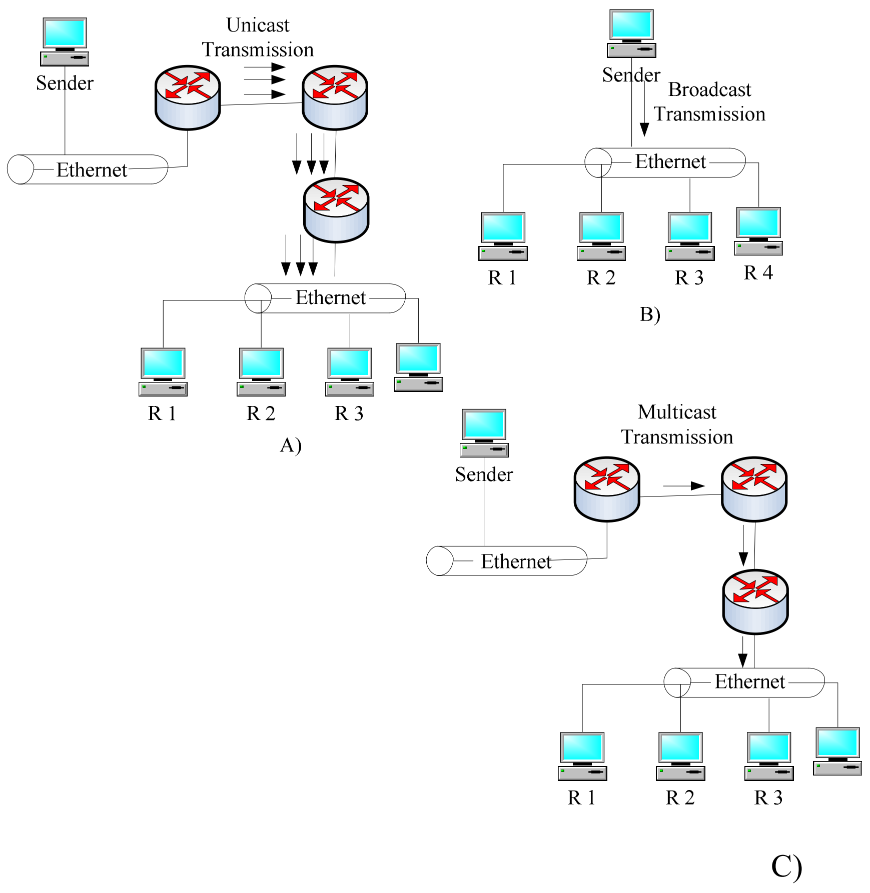


**Figure 2.8 A) Unicast Transmission B) Broadcast Transmission C) Multicast Transmission.**

Although multicast has been very successful at providing an efficient, best-effort data delivery service to large groups, it has proven much more difficulties to extend other features, such as security, in a scalable manner. As compared to unicast, multicast is susceptible to more attacks due to its unique properties. For example, since multicast allows open group membership and open access to send packets to the group [41], it possibly cause eavesdropping, theft of service, or denial-of-service. The precise set of security requirements for group communication is determined by the application using the service. However, a minimal set of requirements can be given that most applications share such as:

### 2.5.1 Source Authentication

Member identification and authentication is essential to prevent an intruder from impersonating a legitimate group member or group manager. In unicast communication, authentication can be achieved in symmetric mechanism: the sender and the receiver share a secret key to compute a message authentication code (MAC) of communication data. After verifying the MAC the receiver is confirmed that the sender generated the message. However, in multicast communication symmetric MAC authentication is not secure, because every receiver knows the MAC key and could impersonate the sender and forge messages to the other receivers. Multicast authentication scheme should uniquely identify the source of each generated message. Some applications requires signature to provide non-repudiation. The addition of digital signature (DS) to each message satisfies both source authentication and non-repudiation.

### 2.5.2 Access Control

After a party has been identified, access control should be performed in order to validate group members before giving them access to group communication data such as group key. Traditional method is to provide decryption keys to the authorized members only. However, there are still risks involved with unauthorized users receiving encrypted data such as traffic analysis or cryptanalysis. Moreover, it is vulnerable to Denial-of-Service (DoS) attack in which malicious member joins a number of multicast groups, utilizing large amounts of bandwidth or router resources. Hence, multicast receiver access control is required to control the ability of hosts to join the multicast group.

### 2.5.3 Data Integrity

Data integrity ensures that the data is not modified or deleted in any unauthorized way. In both unicast and multicast communication, integrity could be achieved through authentication. Authentication ensures that the data originates from the claimed source and the data was not modified during transport.

### 2.5.4 Confidentiality

Confidentiality ensures that the multicast data could be read only by the group members. The typical solution is to encrypt the data with a secret key which is known only by the sender and the legitimate receivers. Distributing a secret key efficiently to a large number of receivers is a challenge. This becomes more complicated when the group membership is dynamic. In most applications, it is necessary to change the group key so that a new member cannot access the old data (known as backward secrecy) and a leaving member cannot access the new broadcast (known as forward secrecy). Moreover, maintaining Group Key secrecy, i.e., ensuring that no outsider can find any group key is also important. The design of an efficient and viable Group Key Management technique is the critical issue for the realization of a secure multicast communication.

### 2.5.5 Watermarking

Encryption is generally used to safeguard content while it is being transmitted so that unauthorized persons cannot read the confidential data. However, this does not offer any protection against unauthorized duplication and propagation by the intended receiver after she receives the data. Watermarking can provide protection in the form of theft deterrence. It embeds some identification information into the content in such a way that it cannot be removed by the user but can be read by the right party.

## 2.6 Multicast over Wireless Network

Multicast over wireless network is defined as the ability to send data to a set of mobile units, communicating to each other for a particular operation, regardless of the mobility of the units [42]. The recent advances in wireless communication technology, wireless devices and mobile workforce, demand for multicast communication over wireless networks. Applications of wireless multicast include group-oriented mobile commerce, military command and control, distance education, intelligent transportation system, mobile auction etc. In military environments, tactical information may be multicast to users, tanks, and planes; distance education and entertainment services can be offered to mobile or remote users; intelligent transportation systems involve the dynamic routing or rerouting of individual vehicles; current traffic information, as well as the most direct and least time-consuming routes, can be multicast to drivers; in the future, commercial aircraft may fly on the most efficient routes guided, in part, by multicasts of location information concerning other nearby aircraft, objects, and destinations [42]. Since the access points of mobile units change over time, it poses several challenges for efficient multicast routing in mobile users [42]. Multicast communication over wireless networks can be divided into two classes: Multicast routing for infrastructure based wireless network and multicast routing for Ad Hoc wireless network [42]. Software distribution in intelligent vehicle is an example of infrastructure based wireless network, whereas inter-vehicle communication for collision avoidance system is an example of Ad hoc network. Infrastructure-based wireless multicast network involves base stations and switches in a fixed topology and mobile users. Existing multicast protocols such as Distance Vector Multicast Routing Protocol (DVMRP), Multicast Extension to Open Shortest Path First (MOSPF), Core-based Tree (CBT), and Protocol Independent Multicast (PIM), can be modified for wireless multicast [42].

## 2.7 Multicast Group Communication Confidentiality and Group Key Management Properties and Proposed Solutions

To ensure confidentiality during multicast session, the sender (source) shares a secret symmetric key with all legitimate group members, named as “Traffic Encryption Key (TEK)” or group “Session Key (SK)”. The sender encrypts a multicast message with the TEK using a symmetric encryption algorithm whereas upon receiving the encrypted message $\{m\}_{TEK}$, each valid receiver that knows the TEK can decrypt it with the TEK and retrieves the message. In order to ensure backward secrecy, forward secrecy or protection against any kind of cryptanalysis, the Group Controller (GC) must generate a new TEK and securely distribute it to the group members. This procedure is known as “re-keying”. The distribution, use and update of keys involved in encryption and re-keying is generally known as “Group Key Management (GKM)”.

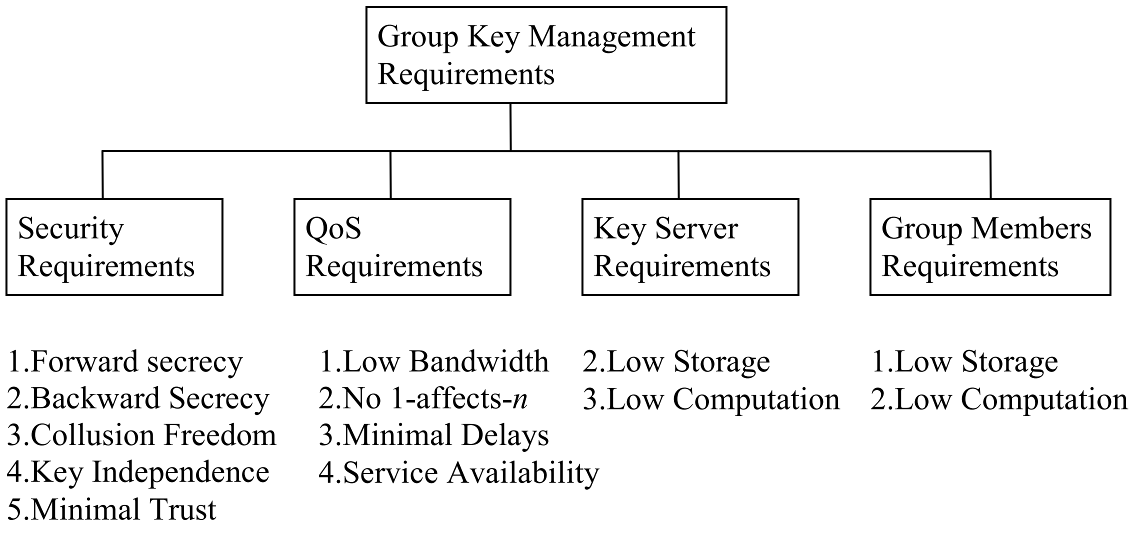


**Figure 2.9 Group Key Management Requirements.**

Figure 2.9 summarizes the requirements of efficient GKM protocols from four points of view [5, 7]:

**Security Requirements**

- ***Forward Confidentiality:*** Users who left the group should not have access to any future key so that they cannot decrypt data after they leaves the group.
- ***Backward Confidentiality*:** A new user that joins the session should not have access to any old key so that it cannot decrypt data sent before it joins the group.
- ***Collusion Freedom*:** Any set of fraudulent users should not be able to deduce the current TEK.
- ***Key Independence*:** The disclosure of a key does not compromise other keys.
- ***Minimal Trust*:** The GKM protocol should not put trust on a high number of entities.

**Quality of Service (QoS) Requirements**

- ***Low Bandwidth Overhead*:** Both the initial keying and re-keying of the group should not incur high number of messages to be exchanged between GC and the group members.
- ***1-affects-n*:** A single membership change in the group should not affect all the other group members.
- ***Minimal Delays*:** The re-keying messages must be delivered in timely manner so that members receive the message before new key takes effect.
- ***Reliability*:** Re-keying messages must be delivered reliably to prevent loss of messages.
- ***Service Availability*:** The failure of a single entity must not stop the whole multicast session.

**Other Requirements**

- ***Low Storage*:** The number of keys that the GC and members need to keep should be low.
- ***Low Computation*:** Computation required by the GC and members to process keying messages should be minimal.

Many secure and scalable key management techniques have been proposed which can be divided into three main categories depending on different features, requirements and goals [5, 7]:

I. **Centralized GKM protocols**: A single key server known as Key Distribution Center (KDC) is responsible for computing and distributing group key to all group members.

II. **Decentralized GKM protocols**: The management of a large group is divided among subgroup managers, thus minimizing the workload in a single place.

III. **Distributed GKM protocols**: There is no explicit KDC, and all the members contribute to generate group key.

## 2.8 Related Works

Several efforts have been taken by the industry and the researchers to create a standard for the ECU interface and communication protocols for RSU. In [43], a generic RSU technique is proposed where one of the ECUs with more processing power and wireless Internet connection capability named as Head Unit (HU) acts as a gateway to receive software from remote maintenance server and distribute it to the targeted ECU via Controller Area Network (CAN). Mutual authentication between the HU and the remote administrative server is assumed to be done using standard security protocols used in WLAN or WWAN. Since all the existing security protocols used in WWLAN and WWAN has some security flaws as stated in Section 2.4, both symmetric and asymmetric/public-key cryptographic techniques and the combination of two (symmetric and asymmetric) have been proposed for secure software download in mobile devices [44]. In symmetric approach, both the SV and mobile terminal share a secret key. The SV uses this key to create a MAC value of the software (e.g., keyed hash function) and send it along with the software. The mobile terminal verifies the authenticity of the SV by checking the MAC and the integrity of the software by comparing the hash value of the received software and the one contained in the MAC. Since both parties share the same secret key, anyone who has the key could generate the MAC, thus it does not guarantee non-repudiation in case of dispute between SV and mobile terminal [44]. Moreover, the secret key must be transmitting to both the parties through secure channel before establishing the software distribution session. Since symmetric-key cryptography is involved in encrypting and decrypting the software, it requires less processing. In public-key approach, it is assumed that a supporting Public-Key-Infrastructure (PKI) exists where trusted parties provide certificates to mobile terminals as well as the SV. The SV generates a license of the software, which contains the information about integrity of the software (cryptographic hash of software), validity date, issuer identity, recipient's identity etc., and digitally signs it. It sends the software and the signed license encrypted with the mobile terminal's public-key. Mobile terminal decrypts the software using its private key and verifies signature using SV's public-key. This technique provides non-repudiation; however, it suffers from high computational cost. Hence, combination of both symmetric and asymmetric techniques, i.e., exchanging a per session secret-key using asymmetric technique and encrypting/decrypting software using symmetric cryptographic technique, provides us with non-repudiation as well as fast encryption/decryption operations. For mutual authentication, authenticated Diffie-Hellman key exchange method [45] can be used to generate a symmetric session key where SV and the mobile terminal possess each other's authentic public key. In VSDN, we assume that vehicles are connected to the software supplier through BS or Intelligent Transportation Tower (ITT). The BS/ITT will receive software from the AC or SV and transmit it to the particular vehicle. Since vehicles are highly mobile device, the authentication process between vehicle and BS/ITT should be fast. Hence, in our architecture we use symmetric cryptographic technique for mutual authentication between vehicle and BS/ITT. In order to prevent the known-key attack, different authentication key is used for different software upload session. Moreover, in order to increase the security level, we propose the vehicle to upload two copies of the software and the message digest (MD) in each copy. An MD algorithm takes signed message as an input and produces a hash which allows to verify the integrity and authenticity of the message. Since the vehicle will not accept the software unless the packets in two copies match, there is no chance that the vehicle will upload the software that is changed by the hacker.

Several GKM protocols for secure multicasting have been proposed in literature mostly for fixed networks [5, 7]. Among these schemes, those that employ tree-hierarchy for the arrangement of keying materials are well-accepted for its communication efficiency. The basic tree-based protocol, known as Logical Key Hierarchy (LKH), has communication complexity $O(\log_2 n)$ for re-keying, group member storage complexity is $O(\log_2 n)$ and GM storage complexity is $O(n)$ when applied to members of $n$ [7]. Recently, cluster-based approaches are adopted for scalable GKM where the key management domain is divided into smaller administratively scoped areas or clusters and the key management role is distributed to each area/cluster head. The area could be logically or physically characterized. Among the proposed protocols, the most popular one is Ious where each cluster head generates the encryption key for the cluster [46]. The cluster head locally controls the re-keying during group membership change, thus keep the other subgroups unaffected and reduce the workload of GM.

Recent advancements in wireless communication technology, wireless devices and mobile workforce, demand for multicast communication over wireless networks. Some applications of wireless multicast are: group-oriented mobile commerce such as mobile auctions, military command and control such as dissemination of tactical information to troops, various ITS applications such as distribution of weather information, optimum route information, traffic information to drivers, software distribution to vehicles' ECUs, inter-vehicle communication for collision avoidance, including mobile users in multicast sessions such as distance education, audio and video conferencing, news distribution, on-demand stock information, pay-per-view game, movie etc. However, host mobility in mobile multicasting introduces additional complexity in GKM. In infrastructure-based wireless mobile multicasting, cluster based decentralized GKM protocols are well-suited where the key management domain is divided into smaller areas/cluster based on the coverage area of a single BS or a group of BSs. The BSs work as cluster heads and may take part in key management depending on the trust model of the system [6]. Following the similar concept, in this thesis we propose two GKM techniques for the multicast group formed for RSU in vehicle ECUs named as FT and ST systems. In the FT system, the BSs have access to software packets and they take part in key management; whereas, in the ST system the BSs act as proxies for the vehicles by honestly relaying the encrypted software packets from the software distribution centers to the vehicles. We describe multicast session establishment methods for these two GKM protocols that comprise authentication of the software distributor and vehicles, establishment of secure link, and generation and distribution of SK to them. Consequently, we developed an analytical and simulation model based on realistic vehicle distribution and movement, wireless channel access mechanism and message reception mechanism to evaluate the performance of these two protocols by finding the multicast session initialization latency and handoff latency. We present numerous analytical and simulation results collected for various distribution of system parameters.

# 3 CHAPTER 3 – SECURE UNICASTING FOR RSU IN A VEHICLE'S ECU

## 3.1 The Proposed Architecture for Secure Software Upload

In the proposed architecture, we assume that the Auto Company (AC) might have its own software distribution center or it has agreement with a third party Software Vendor (SV) to provide the required software. Each vehicle is equipped with several ECUs such as Communication ECU, Crypto ECU, and GPS ECU etc. The communication ECU acts as a gateway to connect to the remote software distribution server via the BS. It is also connected to the local vehicle network (i.e., CAN bus) in order to send the software to the targeted ECU. The AC, the SV and BSs are connected through high-speed wired/wireless networks, whereas the vehicles that travel between cells, can communicate with underlying network via BSs using long-range wireless communication links, e.g., cellular or Wi-Fi links. The BS, under which the targeted vehicle resides, receives software packets from the SV using secure communication technique such as SSL/TLS and transmits the packets to the targeted vehicle through secure wireless link as it is shown in Figure 3.1.

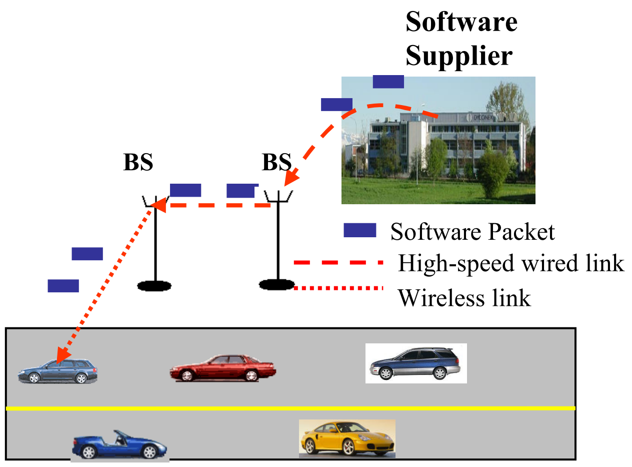


**Figure 3.1 Remote Software Distribution Network using Wireless Link.**

### 3.1.1 Authentication and Key Agreement Process

The crypto-ECU of a vehicle $V_m$ is pre-loaded with a set of $n$ authentication keys $\mathbf{k}_{V_m} = \left(k_{V_m}^1, \cdots, k_{V_m}^n\right)$. Each key is used to authenticate $V_m$ at each software distribution session. A copy of these keys will also be kept in a secure Central Server (CS) which is maintained by the AC or any trusted party. The AC or any other Certification Authority (CA) issues Digital Certificates to the SV and all BSs which contain their authentic public-keys. During authentication process, they send the certificate. We assume that all the vehicles and the BSs have a copy of the SV's authentic public key and the BSs have their neighboring BSs' public key.

When the AC decides to upload software in a vehicle $V_m$, it sends an unused authentication key $k_{V_m}^j$ and the module number to which software needs to be uploaded to the SV using a secure link such as SSL/TLS. Upon receiving the message, the SV creates a *Req_SW_Update_Join* message that consists of a message ID, a Vehicle's ID (VID, could be a part of its VIN number), an ECU ID to which the software needs to be updated, the version number of the software, a session key $k$ , its rekeying period and time stamp. The SV digitally signs it, encrypts the message using $k_{V_m}^{j}$ and sends it to the BS under which $V_m$ is currently located. The BS honestly relays the message to $V_m$. After receiving the *Req_SW_Update_Join* message, $V_m$ decrypts the message using $k_{V_m}^{j}$ , verifies the signature and version number of the software and sends a *Ack_SoftwareUpdate_Join* message. The BS forwards the message to the SV. If authentication fails, the vehicle $V_m$ ignores the message.

### 3.1.2 Sending the Software Packets

After successful authentication of both the vehicle and the SV, the SV starts sending the software packets encrypted with the session key $k$ . The SV can use this key to create a $\mathrm{MAC}$ value of each software packet and send it along with the packet. The vehicle performs the same operation on the software packets to create a new code $\mathrm{MAC}'$ and compares it with the received $\mathrm{MAC}$ to verify the data integrity as it is shown in Figure 3.2. Since person with the key $k$ is only able to generate the $\mathrm{MAC}$ , source authentication is also achieved.

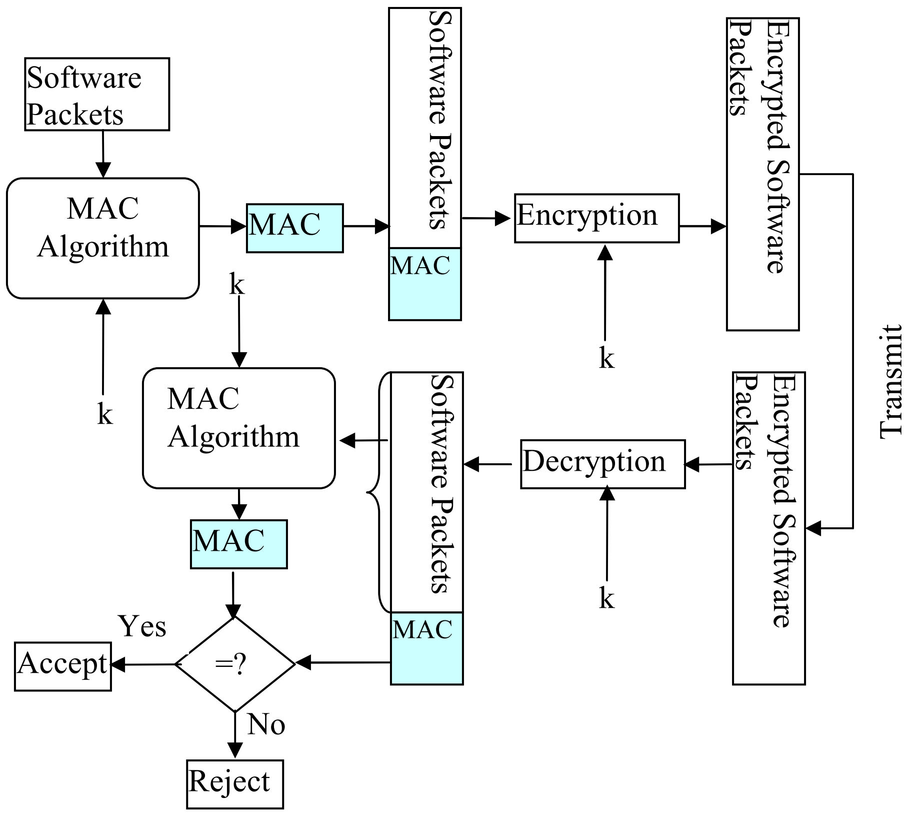


**Figure 3.2 Software Transmission Method.**

Since both parties share the same secret key, anyone who has the key could generate the MAC. Thus, it does not guarantee non-repudiation in case of dispute between the SV and the vehicle. Moreover, if an intruder could successfully change both the packet and the MAC value then there is no way that the vehicle could verify the software. In this thesis, we propose a better software upload technique where the vehicle receives two copies of the software along with the signed Message Digest (MD) in each copy. The MD is an encrypted hash of a message that can take variable length input and produces fixed length output (typically 128 bits or 512 bits). The SV can create a 128-bit MD using MD5 algorithm [47] and encrypts it with its private key. If some packets of the first copy do not match with the corresponding packets in the second copy, the vehicle requests to send the unmatched packets. After receiving both the copies along with the signed MDs, the vehicle calculates an MD based on the received software and compares it with the received MD.  The vehicle accepts the software only when the calculated MD and received MD match.

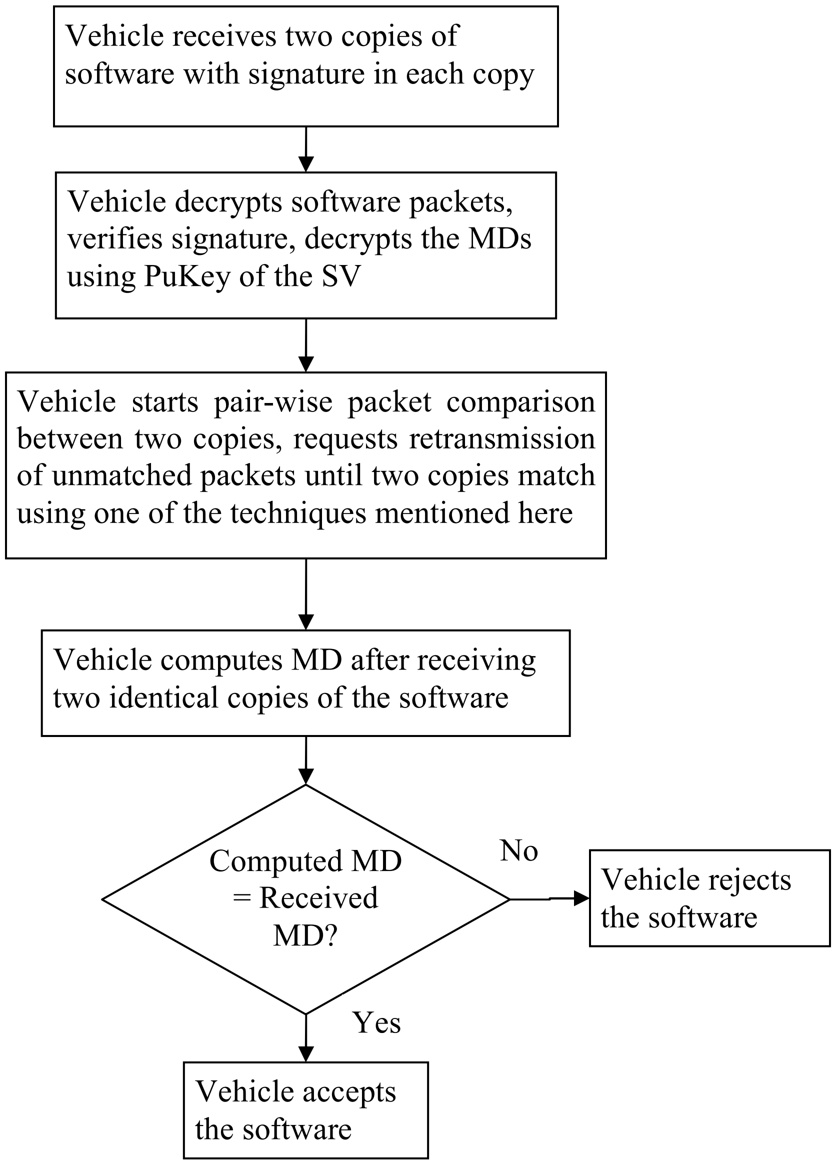

Vehicle receives two copies of software with signature in each copy
Vehicle decrypts software packets, verifies signature, decrypts the MDs using PuKey of the SV
Vehicle starts pair-wise packet comparison between two copies, requests retransmission of unmatched packets until two copies match using one of the techniques mentioned here
Vehicle computes MD after receiving two identical copies of the software
Computed MD = Received MD?
No
Vehicle rejects the software
Yes
Vehicle accepts the software

**Figure 3.3 Two-copy Software Upload Technique.**

Figure 3.3 shows the flow diagram of the technique. In the next section we present several ways how the vehicle receives two copies of the software and find analytical expressions for average number of packet transmissions (*N*) for successful software reception in each case. In order to do the comparison, we also present the expression of *N* for single- copy software upload technique.

## 3.2 System Analysis

In this section, several techniques of how the vehicle receives two copies of the software are presented and analytical expressions are found for average number of packet transmissions (*N*) for successful software reception in each case. In order to do the comparison, we also present the expression of *N* for single- copy software upload technique.

### 3.2.1 Notation

The symbols and notations that will be used throughout the paper are presented in Table 3.1.

TABLE 3.1 NOTATIONS USED IN UNICAST RSU TECHNIQUE.

| **Symbol** | **Significance** |
|---|---|
| $M$ | Total number of software packets without MD |
| $m$ | Number of packets in a segment |
| $S$ | Number of segments = $\frac{M}{m}$ |
| $p$ | Packet error probability due to hacking |
| $p_{pair}$ | Probability that a packet-pair do not match due to hacking |
| $p_{soft}$ | Probability that the received software is in error due to hacking |

| | |
|---|---|
| $T$ | Average number of trial to send one packet or one segment or the total software successfully |
| $p_i$ | Probability of success in $i^{th}$ trial |
| $N_p$ | Average number of packet transmission to receive one good packet |
| $N$ | Average number of packet transmission for successful software upload |

### 3.2.2 Definitions

Figure 3.4 shows different software upload techniques that we consider in the analysis.

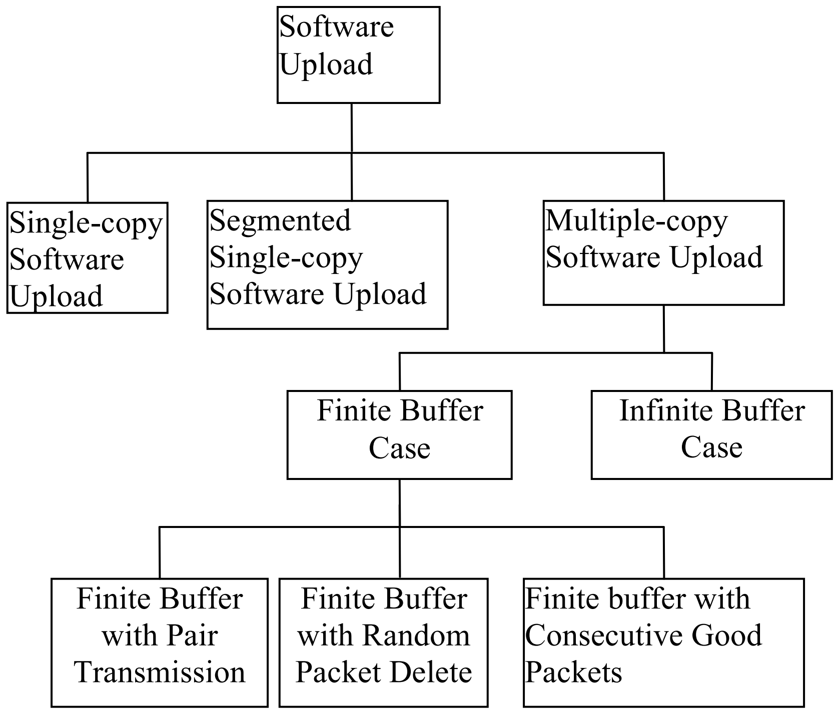

Software Upload
Single-copy Software Upload
Segmented Single-copy Software Upload
Multiple-copy Software Upload
Finite Buffer Case
Infinite Buffer Case
Finite Buffer with Pair Transmission
Finite Buffer with Random Packet Delete
Finite buffer with Consecutive Good Packets

**Figure 3.4 Different Unicast Software Upload Techniques.**

**Single-copy Software Upload**

If there is only one buffer in vehicle's software module to accept the new software and one copy of the software packets is sent appended with the MD then it is called *Single-copy Software Upload*.

**Segmented Single-copy Software Upload**

If the software packets are divided into segments of certain number of packets and each segment is sent with the MD then it is called the *Segmented Single-copy Software Upload.*

**Multiple-copy Software Upload**

If there are more than one buffer and multiple copies of the software are sent with the MD in each copy until there is a match found then it is called the *Multiple-copy Software Upload.*

**Infinite Buffer Case**

If there are infinite number of buffers to accept multiple copies of the software to compare a new copy with the already received copy until a match is found then it is called the *Infinite Buffer Case*. This is the ideal case and not practical, which requires minimum number of packet transmissions for a successful software upload.

**Finite Buffer Case**

If there are two buffers to accept two copies of a packet and one or both of the packets are replaced by the new packets transmitted until the vehicle receives a good packet then it is called the *Finite buffer Case*.

**Finite Buffer with Pair Transmission**

If a packet-pair do not match then the vehicle could delete both packets and request to send another pair until a matched pair is found. This case is defined as the *Finite Buffer with Pair Transmission*.

**Finite Buffer with Random Packet Delete**

If a packet-pair do not match then the vehicle could delete one randomly chosen packet and request to send another packet until a matched pair is found. This case is defined as the *Finite Buffer with Random Packet Delete*.

**Finite Buffer with Two Consecutive Good Packets**

If a packet-pair do not match then the vehicle always deletes the older packet and requests to send another packet until a matched pair is found. This case is defined as the *Finite Buffer with Two Consecutive Good Packets*.

### 3.2.3 Single-copy Software Upload

After receiving all the encrypted software packets and the signature, the receiving vehicle decrypts the packets, verifies the signature, calculates an MD and compares it with the received MD. If both the MDs match, then the vehicle accepts the software. Otherwise, it requests the supplier to retransmit the entire software. In this method, if a hacker changes at least one software packet, then the calculated MD will differ from the received MD. Since the vehicle or the supplier does not know which packet has been changed, the supplier needs to retransmit the entire software including the MD which requires more network bandwidth. Moreover, if a hacker can successfully change a packet from every transmission, it is not possible at all to upload the software successfully.

For packet error probability $p$ due to hacking, the probability that the software is in error is:

$$p_{soft} = 1 - (1 - p)^{M+1} \tag{3.1}$$

The average number of trials required to send the software successfully is

$$T = (1 - p_{soft}) \sum_{i=1}^{\infty} i p_{soft}^{i-1} = \frac{1}{1 - p_{soft}} = \frac{1}{(1 - p)^{M+1}} \tag{3.2}$$

The average number of packets transmission for successful software upload is

$$N = (M + 1)T = \frac{M + 1}{(1 - p)^{M+1}} \tag{3.3}$$

## 3.2.4 Segmented Single-copy Software Upload

In case of Single-copy Transmission, if the number of software packets $M$ increases, the average number of packet transmission for successful software upload increases exponentially as shown in eq. (3.3). An alternative approach could be to divide $M$ software packets into $S$ segments with $m$ packets in each segment. Then the average number of trials required sending one segment successfully is

$$T = \frac{1}{(1 - p)^{m+1}} \tag{3.4}$$

Average number of packet transmission needed for successful upload of $S$ segments is

$$N = (m + 1)ST = \frac{(m + 1)S}{(1 - p)^{m+1}} \tag{3.5}$$

### 3.2.5 Multiple-copy Software Upload

#### Infinite Buffer Case

For each software packet, the vehicle first receives two copies of the packet. If the packets do not match, it requests to send another copy of the packet. The third copy is compared with the previous two. If no match is found it requests for another copy. Since there is infinite number of buffers, after receiving $i^{th}$ packet it compares the packet with previous $i-1$ packets. The process continues until a matched-pair is found. The probability that a packet is received successfully in the $i^{th}$ trial is

$$P_i = ip^{i-1}(1-p)^2, \; i = 1,2,3\cdots,\infty \tag{3.6}$$

The average number of packet transmission for successful upload of one packet is

$$N_p = \sum_{i=1}^{\infty}(i+1)P_i = \frac{2}{1-p} \tag{3.7}$$

The average number of packet transmission for successful software upload is

$$N = (M+1)N_p = \frac{2(M+1)}{1-p} \tag{3.8}$$

#### Finite Buffer with Pair Transmission

In this case, if both the copies of a packet do not match, the supplier will send another pair of packets.

The probability that a pair does not match is

$$p_{pair} = 1-(1-p)^2 \tag{3.9}$$

The average number of trials to send one packet successfully is

$$T = \sum_{i=1}^{\infty} i(1-p)^2 \left(1-(1-p)^2\right)^{i-1} = \sum_{i=1}^{\infty} i(1-p_{pair})p_{pair}^{i-1}$$

$$\Rightarrow T = \frac{1}{1-p_{pair}} = \frac{1}{(1-p)^2} \tag{3.10}$$

The average number of packet transmissions for successful software upload is

$$N = 2(M+1)T = \frac{2(M+1)}{(1-p)^2} \tag{3.11}$$

**Finite Buffer with Two Consecutive Good Packets**

When the two received copies of a packet do not match, the vehicle replaces the first copy in buffer 1 with the second copy in buffer 2, requests to send another copy and places in buffer 2. The average number of packet transmissions for successful upload of one packet is

$$N_p = \sum_{i=1}^{\infty} (i+1)P_i = \frac{2-p}{(1-p)^2} \tag{3.12}$$

where $P_i$ is the probability that a packet is received successfully in the $i^{th}$ trial.

Then the average number of packet transmissions for successful software upload is

$$N = (M+1)N_p = \frac{(M+1)(2-p)}{(1-p)^2} \tag{3.13}$$

# 4 CHAPTER 4 – SECURE MULTICASTING FOR RSU IN VEHICLES' ECUS

## 4.1 Problem Statement

Implementing multicast communication in VSDN poses several challenges due to many reasons such as:

1. It consists of highly mobile vehicles that results in frequent handoff and group membership change of the MG. Group members moving from one cell to another may require synchronized transfer of keying materials between the leaving and entering BSs.
2. If only the GM (the AC or the SV) is responsible for GKM, then it could be susceptible to single-point failure.
3. Broadcast nature of wireless links lacks control on wireless receivers and poses more risks of being eavesdropped thus introduce additional complexity in GKM.
4. Wireless network has different channel characteristics and mobility dynamics that make network design and analysis more challenging.
5. There is limited and variable amount of bandwidth available in both directions resulting in inefficient multicast tree and incorrect routing.
6. Frequent handoff may cause more packet loss and network delay.
7. For infrastructure-based mobile multicasting putting too much trust on BSs will reduce the security of the multicast network.

## 4.2 Design Objectives

This work aims to achieve the following objectives while designing the GKM protocols:

1. **Mutual Authentication and non-repudiation**: Vehicles should accept software only from legitimate SV. Similarly, the SV should deliver the software to the targeted vehicles only. Hence, mutual authentication is needed before transmitting the software to vehicle ECUs. In case of dispute, the SV should be reliably identified.
2. **Low communication overhead**: Multicast session initialization and handoff should not incur high number of messages to be exchanged between the GM and the group members.
3. **Minimal trust**: The GKM protocol should not put trust on a large number of entities.
4. **Scalability**: The scheme should scale well to a large number of receivers.
5. **Single point of failure**: The scheme should not rely on a single entity for GKM. Otherwise, it would be susceptible to single-point failure.
6. **Low computation**: Computation required by the GM and members to process keying message should be minimal.
7. **Low storage**: The number of keys that the GM and members need to store should be low.

## 4.3Assumptions

In order to facilitate GKM protocols, we make the following assumptions:

1. The large multicast group that consists of the SV and the vehicles is divided into small subgroups based on region and a Regional Group Manager (RGM) is responsible for distributing the software to the vehicles located in its region.
2. The VSDN consists of BSs and vehicles. Hence, the network dynamic is characterized by quasi-permanent mobility and high speed. The vehicle trajectories are defined by the roads. Since the road trajectory is known, the number of MG members (vehicles) under a BS is predictable and there is a limit of maximum number of group members in an MG. Hence, scalability is not a big concern in VSDN.
3. When a vehicle joins (leaves) a new (current) BS, it might join (leave) a new (current) MG. However, the new (leaving) member is a vehicle from the same company and the software needs to be uploaded in that vehicle. Hence, there is no need to update the SK when a vehicle joins (leaves) an MG. Rather, the SK needs to be delivered to the joining vehicle. In other words, there is no need for forward and backward secrecy. However, periodic key refreshment could be done to protect the system security against cryptanalysis attack.
4. Since the Auto Company will initiate the software update session, the Sender-initiated [48] multicast protocol is suitable for this application contrary to receiver-initiated group communication in traditional multicast.
5. All the entities involved in the multicast architecture are time synchronized using an appropriate time synchronization protocol such as Network Time Protocol (NTP) or Global Positioning System (GPS).

6. Unlike other mobile devices such as PDA, vehicles provide sufficient computational and power resources. Moreover, exponential increase in processing power and related improvements in computational platform will make the vehicle’s processor more powerful. This allows vehicle’s processor to compute complex cryptographic algorithms.

## 4.4 Proposed Multicast Architecture for RSU

The proposed multicast architecture for RSU hierarchically consists of the AC or the ITS authorized software vendor named Central Manager (CM), its regional offices known as RGM, the BSs owned by the ITS or other third party wireless vendor and vehicles to which software needs to be uploaded (Figure 4.1). The CM administers the RGMs; each RGM controls the BSs residing in its region, and each BS manages the vehicles under its cell. The CM, RGMs and BSs are connected through high speed wired/wireless links, whereas vehicles can communicate with their respective BSs using wireless communication links, e.g., cellular, Wi-MAX, IEEE 802.20 or Wi-Fi. The software packets are first routed to the RGMs from the CM using

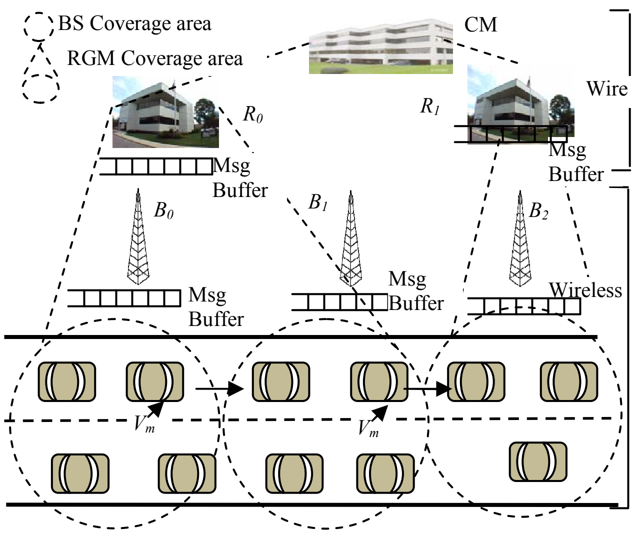


**Figure 4.1 The Multicast Architecture for RSU.**

secure channels such as Transport Layer Security (TLS) protocol. In case of the FT system, each RGM forms an IP-based reliable and secure multicast group with its underlying BSs named *RGM-BS* multicast group and transmits the packets to the BSs. Again, the nearby BSs under an RGM form clusters and each cluster along with the vehicles under it forms a *BS-vehicle* group named as *Multicast Group* (MG). The MG is divided into two subgroups: *Data Group* (DG) that comprises all the vehicles and *Control Group* (CG) that comprises all the BSs in an MG. Upon successful reception of software packets from the RGM, each CG forwards the packets to the DG using secure and reliable wireless multicasting. In case of the ST system, the CG is not trusted. Hence, the RGM sends the packets to DG using CG as proxy to reliably forward the packets to DG.

## 4.5 Group Key Management Protocols

### 4.5.1 Notation

The symbols and notation that will be used throughout the paper is presented in Table 4.1.

TABLE 4.1 NOTATION USED IN GKM PROTOCOLS FOR WIRELESS MULTICASTING.

| | |
|---|---|
| $\mathbf{R} = \{R_1, \cdots, R_{N_R}\}$ | Set of RGMs owned by the Auto Company or ITS with cardinality $N_R$ |
| $\mathbf{B} = \{B_1, \cdots, B_N\}$ | Set of BSs owned by the ITS or other 3rd party wireless vendor with cardinality $N$ |
| $\mathbf{B}^i = \{B_1^i, \cdots, B_{N_B^i}^i\}$ | Set of BSs under an RGM $R_i$ with cardinality $N_B^i$ |
| $\mathbf{V}^i = \{V_1^i, \cdots, V_{N_V^i}^i\}$ | Set of vehicles under an RGM $R_i$ with cardinality $N_V^i$ |
| $\mathbf{CG}^i = \{CG_1^i, \cdots, CG_{N_{CG}^i}^i\}$ | Set of Control Groups under an RGM $R_i$ with cardinality $N_{CG}^i$ |
| $N_B$ | Number of BSs in a CG or MG |
| $\mathbf{M}^i = \{M_1^i, \cdots, M_{N_M^i}^i\}$ | Set of MG/CG under an RGM $R_i$ with cardinality $N_M^i = \dfrac{N_B^i}{N_B}$ |
| $\mathbf{B}_m^i = \{B_{(m-1)N_B+1}^i, \cdots, B_{m.N_B}^i$ | Set of BSs within an MG under an RGM $R_i$ where $m = 1, \cdots, N_M^i$ |
| $B_{L_m}^i$ | Group leader BS of a CG/MG $M_m^i$ |
| $\mathbf{V}_{M_j^i} = \left\{V_1, \cdots, V_{N_{VM_j^i}}\right\} \subseteq \mathbf{V}$ | Set of vehicles within a DG or MG |
| $\mathbf{V}_{B_j^i} = \left\{V_1, \cdots, V_{N_{V_j^i}}\right\} \subseteq \mathbf{V}_{M_j^i}$ | Set of vehicles under a BS $B_j^i$ |
| $T_k$ | Re-keying period of key *k* |
| $k'$ | Updated key for key *k* |
| $k_{V_m}^l$ | Authentication key of $V_m$ for $l$th software update session |
| $Ek$ | Symmetric encryption with key *k* |
| $Sign_A$ | Digital Signature of entity A |
| $P_A$ | Authentic public key of an entity *A* |
| $P_A^{-1}$ | Private key of an entity *A* |
| $\forall$ | For all |
| $\in$ | In |

### 4.5.2 RGMs, BSs and Vehicles Authentication Keys and Digital Certificates

We assume that the AC or any other Certification Authority (CA) issues Digital Certificates to all the RGMs and BSs which contain their authentic public keys, the CA's public key and the validity period of the certificate. The CA provides them with new certificates when the validity period is about to expire. Each RGM (BS) pair-wise share a secret key with its respective authority i.e., CM (RGM) known as *RGM (BS) Authentication Key*. This key could be pre-distributed or established during the session initialization phase using Diffie–Hellman key establishment protocol secured with the signature scheme [21, 45, 49]. Each vehicle is equipped with several ECUs such as Communication ECU, Crypto ECU, and GPS ECU etc. The Crypto ECU is pre-loaded with a set of keys known as *Vehicle Authentication Keys*. Both the CM's secure server and the regional office's secure server, to which the vehicle is registered, keep a copy of the keys. Each key is used to authenticate the vehicle at each software distribution session. When all the keys are used, the RGM sends another set of key.

### 4.5.3 Multicast Session Initialization

Multicast session initialization comprises the authentication of CM, RGMs, BSs and the vehicles; generation of a SK for an MG and secure distribution of the SK to the group members. Since CM, RGMs and BSs are fixed and construct the wired part of the network, they can easily be authenticated by their respective authority using any standard mutual authentication technique. The authentication of vehicles and distribution of SK to them makes the non-trivial part of the network. Hence, we describe the vehicle authentication process and distribution of SK to them below.

#### 4.5.3.1 Fully-Trusted System

In the FT system, we assume that each CG in an MG decides on a group leader ($B^i_{L_m}$) who is responsible for generating the SK for that MG. The group leader could be randomly selected by the RGM. The group leader $B^i_{L_m}$ authenticates itself using a standard authentication protocol and establishes a shared secret with other group members. It generates a CG key $k^i_{CG_m}$ and distributes it to other members during the authentication process. After successful authentication, it generates multicast SK, its rekeying period and multicasts it to each member in a CG encrypted with $k^i_{CG_m}$ along with its signature. Each BS in a CG verifies the signature, retrieves the SK and constructs a *Req_DG_Join* message that consist of its own ID, the RGM's ID to which it belongs to, a multicast group ID, a message ID, a VID, an ECU ID, the version number of the software, time stamp and its digital certificate. It digitally signs the message and broadcasts it along with the signature.

$B^i_j \Rightarrow \mathbf{V}_{B^i_j}$ : $\mathrm{Re}q\_DG\_Join, Sign_{B^i_j}(H)$ where $H$ is the hash of the message.

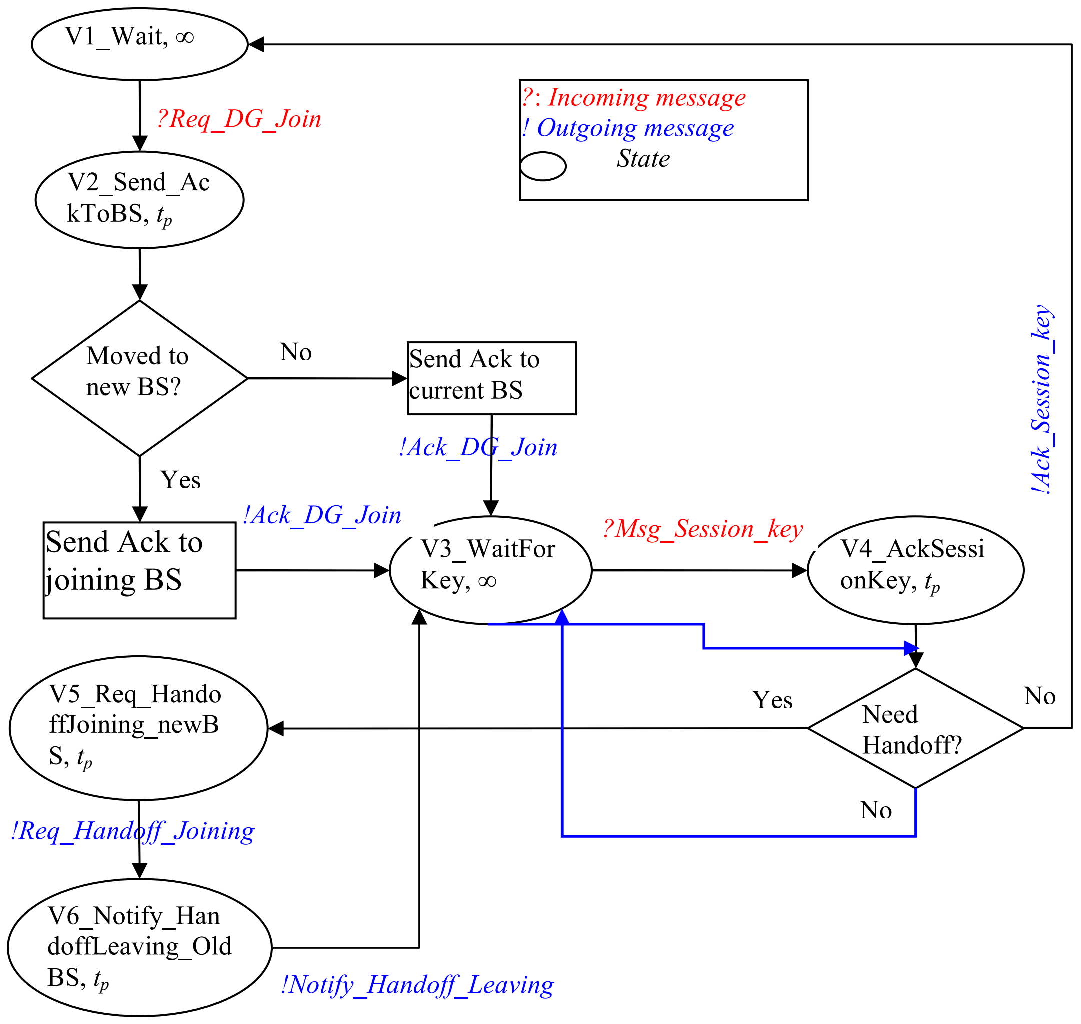


**Figure 4.2 State Transition Diagram of Vehicle during Multicast Session Establishment for both the FT and ST Systems.**

Since a vehicle under a BS may be moving while receiving the *Req_DG_Join* message, it may need hand-off any time. We developed a robust algorithm that can handle any number of handoffs that may be performed by a vehicle during multicast session establishment which is described below. Figure 4.2 shows the vehicle's state transition diagram during multicast session establishment for both the FT and ST systems.

Whenever handoff is needed, the vehicle sends *Req_Handoff_Joining* message to the joining BS and *Notify_Handoff_Leaving* message to the leaving BS.

1. Suppose a vehicle $V_m$ of VID type is under the BS $B_0$ while receiving *Req_DG_Join* message as shown in Figure 4.1. It verifies the signature, timestamp and version number of the software to determine whether it should join the multicast session. If it is still under $B_0$, it sends *Ack_DG_Join* message to $B_0$ that includes its VIN. Else if it moves to $B_1$, it sends *Ack_DG_Join* message to the joining BS $B_1$ mentioning that it received the *Ack_DG_Join* message from $B_0$. After that it goes to *V3_WaitForKey* state to wait for multicast SK from the BS.

$V_m \to B_0(B_1) : Ack\_DG\_Join$

2. While $V_m$ is in *V3_WaitForKey* state, if handoff is needed it sends *Req_Handoff_Joining* message, consists of its VIN, message ID, leaving BS ID to the joining BS and *Notify_Handoff_Leaving* message, consists of its VIN, message ID, joining BS ID to the leaving BS. It returns to the *V3_WaitForKey* state.

$V_m \to B_1(B_2) : \mathrm{Re}q\_Handoff\_Joining$

$V_m \to B_0(B_1) : Notify\_Handoff\_Leaving$

3. When $B_0$ ( $B_1$ ) starts processing the *Ack_DG_Join* message sent by $V_m$, it checks its buffer to verify whether $V_m$ has already sent handoff notification. Accordingly, it determines the current location of $V_m$ (i.e., current BS and RGM under which $V_m$ resides). The $B_0$ ( $B_1$ ) forwards $V_m$'s VIN and current location to its corresponding RGM $R_0$.

$B_0(B_1) \rightarrow R_0 : Ack\_DG\_Join\_BSToRGM$

4. $R_0$ saves the current location of $V_m$ , uses $V_m$ 's VIN to find the next unused authentication key $k_{V_m}^j$ either from its own database or from the CM.

$R_0 \rightarrow CM : Ack\_DG\_Join\_RGMToCM$

$CM \rightarrow R_0 : Ek_{R_0}\left(Msg\_Vehicle\_Auth\_Keys\_CMToRGM\right)$

5. If $V_m$ resides under a BS which is administered by $R_0$ (e.g., $B_0$ or $B_1$ in (Figure 4.1), $R_0$ transfers $k_{V_m}^j$ to the corresponding BS ( $B_0$ or $B_1$ ) encrypted with that BS's shared secret with $R_0$. Otherwise, it follows step 7.

$R_0 \rightarrow B_0(B_1) : Ek_{B_0(B_1)}\left(Msg\_Vehicle\_Auth\_Keys\right)$

6. After retrieving $k_{V_m}^j$ , $B_0$ ( $B_1$ ) again checks its buffer for any handoff notification from $V_m$ . If not found,

   a. $B_0$ ( $B_1$ ) sends *Msg_Session_key* message to $V_m$ that contains multicast group Id of the MG to which it belongs to, multicast SK and its re-keying period, encrypted with $k_{V_m}^j$ . It also contains BS's digital certificate if $V_m$ joins the BS through handoff.

      $B_0(B_1) \rightarrow V_m : Ek_{V_m}^j\left(Msg\_Session\_Key\right)$

   b. $V_m$ retrieves the SK by decrypting the message using the same key $k_{V_m}^j$ and sends *Ack_Session_Key* encrypted with $k_{V_m}^j$ to $B_0$ ( $B_1$ ) if it is still under the coverage area of $B_0$ ( $B_1$ ).

$V_m \to B_0(B_1): Ek^j_{V_m}(Ack\_Session\_Key)$

Else, $V_m$ notifies handoff to $B_0$ ( $B_1$ ). After receiving handoff message from $V_m$, $B_0$ ( $B_1$ ) follows step 8.

c. After verifying *Ack_Session_Key* message received from $V_m$, $B_0$( $B_1$ ) adds $V_m$ to its DG list.

7. Else if $V_m$ resides under a BS which is administered by another RGM, e.g., $B_2$ in Figure 4.1 administered by $R_1$, $R_0$ transfers $k^j_{V_m}$ to the joining RGM $R_1$ encrypted with $R_1$ 's public key. Consequently, $R_1$ sends $k^j_{V_m}$ to $B_2$ encrypted with $B_2$ 's shared secret with $R_1$ and $B_2$ follows step 6.

$R_0 \to R_1 : (Msg\_Vehicle\_AuthKey\_From\_LeavingRGM)_{P_{R_1}}, Sign_{R_0}(H)$

8. The leaving BS $B_0$( $B_1$ ) transmits $k^j_{V_m}$ to the joining BS ( $B_1$ ) if $V_m$ is still under the same RGM ( $R_0$ ) encrypted with the receiver's public key and digital signature.

$B_0(B_1) \to B_1(R_1): (Msg\_Vehicle\_AuthKey\_From\_LeavingBS)_{P_{B_1}(P_{R_1})}, Sign_{B_0(B_1)}(H)$

Otherwise, it forwards the handoff message to leaving RGM ( $R_0$ ) and $R_0$ follows step 7. Ultimately, the joining BS receives the authentication key and follow step d to authenticate $V_m$ and provide it with the SK.

#### 4.5.3.2 Semi-trusted System

Since in the ST system BSs are not trusted, the RGM authenticates the vehicles and provides them with the SK. Multicast session initialization steps are:

1. Each RGM generates session keys and their rekeying period for all the MGs administered by it. It generates *Req_DG_Join* message, signs it and broadcasts to the underlying BSs.

   $R_i \to B_j^i : \mathrm{Re}q\_DG\_Join, \quad Sign_{R_i}(H) \quad for \quad \forall B_j^i \in \mathbf{B}^i, \quad j = 1, 2, 3, \cdots, N_B^i$

2. Each BS forwards this message to the vehicles residing its cell.

   $B_j^i \Rightarrow \mathbf{V}_{B_j^i} : \quad \mathrm{Re}q\_DG\_Join, Sign_{R_i}(H)$

3. Using the similar procedure as it is in FT system, a vehicle $V_m$ under the BS $B_0$ (Figure 4.1) sends *Ack_DG_Join* message to $B_0$ or $B_1$ depending on whether $V_m$ needs handoff or not and goes to *V3_WaitForKey* state . $B_0(B_1)$ forwards this message to $R_0$.

   $V_m \to B_0(B_1) : Ack\_DG\_Join$

   $B_0(B_1) \to R_0 : Ack\_DG\_Join$

4. While $V_m$ is in *V3_WaitForKey* state, if handoff is needed it sends *Req_Handoff_Joining* message, to the joining BS and *Notify_Handoff_Leaving* message to the leaving BS. It returns to the *V3_WaitForKey* state.

   $V_m \to B_1(B_2) : \mathrm{Re}q\_Handoff\_Joining$

   $V_m \to B_0(B_1) : Notify\_Handoff\_Leaving$

The BS $B_0$ ( $B_1$ ) forwards the handoff leaving message to $R_0$. Similarly, $B_1$ ( $B_2$ ) forwards handoff joining message to the RGM $R_0(R_1)$.

$B_0(B_1) \to R_0 : Notify\_Handoff\_Leaving$

$B_1(B_2) \rightarrow R_0(R_1) : \mathrm{Re}\,q\_Handoff\_Joining$

5. When $R_0$ starts processing *Ack_DG_Join* message sent by $V_m$, it checks its' buffer for any handoff notify message related to $V_m$. Accordingly, it finds the current location of $V_m$. $R_0$ finds $V_m$'s authentication key $k^j_{V_m}$ as it is in the FT system either from it's own database or from the CM.

   $R_0 \rightarrow CM : Ek_{R_0}(Ack\_DG\_Join\_RGMToCM)$

   $CM \rightarrow R_0 : Ek_{R_0}(Msg\_Vehicle\_Auth\_Keys\_CMToRGM)$

6. If $V_m$ resides under a BS within its region (e.g., $B_0$ or $B_1$), $R_0$ sends the SK of the related MG to $B_0(B_1)$ encrypted with $k^j_{V_m}$. Otherwise, it follows step 10.

   $R_0 \rightarrow B_0(B_1) : Ek^j_{V_m}(Msg\_Session\_Key)$

7. Meanwhile, if $B_0(B_1)$ receives handoff notify message from $V_m$, it forwards it to $R_0$ as it is in step 4 and $R_0$ takes action as it is in step 6. Also, $B_0(B_1)$ forwards the SK to $V_m$.

   $B_0(B_1) \rightarrow V_m : Ek^j_{V_m}(Msg\_Session\_Key)$

8. $V_m$ decrypts the SK and sends *Ack_Session_Key* encrypted with $k^j_{V_m}$ to $B_0(B_1)$ if it is still under the coverage area of $B_0(B_1)$. Otherwise, it notifies handoff to the leaving BS and the leaving BS forwards it to the respective RGM. The RGM follows step 6.

   $V_m \rightarrow B_0(B_1) : Ek^j_{V_m}(Ack\_Session\_Key)$

9. $B_0(B_1)$ forwards the *Ack_Session_Key* message to $R_0$, $R_0$ decrypts and verifies it. If verification is successful, $R_0$ adds $V_m$ to the right DG group. $R_0$ also sends a message to $B_0(B_1)$ to register $V_m$ to its list of vehicles.

   $B_0(B_1) \rightarrow R_0 : Ek^j_{V_m}(Ack\_Session\_Key\_BSToRGM)$

10. If $V_m$ 's current location is under a BS (e.g., $B_2$) which is administered by another RGM (e.g., $R_1$), the leaving RGM $R_0$ sends $k^j_{V_m}$ to the joining RGM $R_1$ encrypted with $R_1$ 's public key. $R_1$ follows step 6.

$R_0 \rightarrow R_1 : (Msg\_Vehicle\_AuthKey\_From\_LeavingRGM)_{P_{R_1}}, Sign_{R_0}(H)$

### 4.5.4 Hand-off during Multicast Session

We categorize handoff as three different types: a *BS-level handoff* is performed when a vehicle moves from one cell to another within the same MG, an *MG-level handoff* is performed when a vehicle moves from one cell to another under a different MG managed by the same RGM and an *RGM-level handoff* is performed when a vehicle moves from one cell to another under different RGM.

#### 4.5.4.1 Fully-trusted System

1. While receiving multicast data if a vehicle $V_m$ detects that it is entering into a new cell (by comparing signal strength or any other standard method), it sends *Req_Handoff_Joining* message to the joining BS ($B_1$ or $B_2$) in response to the beacon signal received from the joining BS. The *Req_Handoff_Joining* message

consist of its VIN, the multicast group ID, the RGM ID, the old BS's ID and the sequence number of the last message it received. It also sends *Req_Handoff_Leaving* message to the leaving BS ($B_0$ or $B_1$) that contains its VIN and joining BS's ID encrypted with $k_{V_m}^j$.

$V_m \rightarrow B_1(B_2): \mathrm{Re}q\_Handoff\_Joining$

$V_m \rightarrow B_0(B_1): Ek_{V_m}^j(Notify\_Handoff\_Leaving)$

2. The leaving BS $B_0$ transfers $k_{V_m}^j$ to $B_1$ encrypted with $B_1$'s public key along with its digital signature. The joining BS $B_1$ sends *Ack_Handoff_Joining* message to $V_m$ that consist of its ID, public key and time stamp. If it's an *MG-level* handoff then, the *Ack_Handoff_Joining* also contains the SK and its expiration period. Upon receiving the SK, $V_m$ sends *Ack_Session_Key* message to $B_1$. In both cases, $B_0$ deletes $V_m$ from its member list and $B_1$ adds it to the list.

   $B_0 \rightarrow B_1 : (Msg\_Vehicle\_AuthKey\_From\_LeavingBS)_{P_{B_1}}, Sign_{B_0}(H)$

   $B_1 \rightarrow V_m : Ek_{V_m}^j(Ack\_Handoff\_Joining)$

   $V_m \rightarrow B_1 : Ek_{V_m}^j(Ack\_Session\_Key)$

3. If its an *RGM-level* handoff then the leaving BS $B_1$ forwards the handoff notification message to its RGM $R_0$. $R_0$ sends *Msg_Vehicle_AuthKey_From_LeavingRGM* message to $R_1$ encrypted with $R_1$'s public key along with its signature. $R_1$ forwards $k_{V_m}^j$ to $B_1$ follows step 2.

$B_1 \rightarrow R_0 : Ek_{B_1}(Notify\_Handoff\_Leaving)$

$R_0 \rightarrow R_1 : (Msg\_Vehicle\_AuthKey\_From\_LeavingRGM)_{P_{R_1}}, Sign_{R_0}(H)$

$R_1 \rightarrow B_2 : Ek_{B_2}(Msg\_Vehicle\_AuthKey)$

$B_2 \rightarrow V_m : Ek^j_{V_m}(Ack\_Handoff\_Joining)$

$V_m \rightarrow B_2 : Ek^j_{V_m}(Ack\_Session\_Key)$

Note that, while waiting for response from the joining BS, $V_m$ might move to new cell. That is why before sending *Ack_Handoff_Joining* message, the joining BS checks its buffer for handoff leaving message from $V_m$. If there is any then it follows step 2.

### 4.5.4.2 Semi-trusted System

1. Similar to FT-system, the vehicle $V_m$ sends *Req_Handoff_Joining* message to the joining BS ($B_1$) and *Req_Handoff_Leaving* message to the leaving BS ($B_0$). In all cases, the joining and leaving BSs forward the messages to their corresponding RGMs.

   $V_m \rightarrow B_1(B_2) : \mathrm{Re}q\_Handoff\_Joining$

   $V_m \rightarrow B_0(B_1) : Ek^j_{V_m}(Notify\_Handoff\_Leaving)$

   $B_1(B_2) \rightarrow R_0(R_1) : \mathrm{Re}q\_Handoff\_Joining$

   $B_0(B_1) \rightarrow R_0 : Ek^j_{V_m}(Notify\_Handoff\_Leaving)$

2. If it is a *BS*- or *MG-level* handoff, then the RGM $R_0$ sends *Ack_Handoff_Joinig* message to $V_m$ encrypted with $k_{V_m}^j$ via the joining BS $B_1$ and $V_m$ sends *Ack_Session_Key* message to the RGM via $B_1$.

   $R_0 \to B_1 : Ek_{V_m}^j(Ack\_Handoff\_Joining)$

   $B_1 \to V_{1m} : Ek_{V_m}^j(Ack\_Handoff\_Joining)$

   $V_m \to B_1 : Ek_{V_m}^j(Ack\_Session\_Key)$

   $B_1 \to R_0 : Ek_{V_m}^j(Ack\_Session\_Key)$

3. If it is an *RGM-level* handoff then the leaving RGM $R_0$ sends *Msg_Vehicle_AuthKey_From_LeavingRGM* message to $R_1$ encrypted with $R_1$'s public key along with its signature. $R_1$ verifies the message, checks it buffer to see whether $V_m$ already moved away from the cell. If yes, then $R_1$ follows step 2. Otherwise, $R_1$ sends *Ack_Handoff_Joining* message to $V_m$ that contains the SK of new MG encrypted with $k_{V_m}^j$ via $B_1$. The $V_m$ sends *Ack_Session_Key* message to the RGM via $B_1$.

   $R_0 \to R_1 : (Msg\_Vehicle\_AuthKey\_From\_LeavingRGM)_{P_{R_1}}, Sign_{R_0}(H)$

   $R_1 \to B_2 : Ek_{V_m}^j(Ack\_Handoff\_Joining)$

   $B_2 \to V_m : Ek_{V_m}^j(Ack\_Handoff\_Joining)$

   $V_m \to B_2 : Ek_{V_m}^j(Ack\_Session\_Key)$

$B_2 \to R_1 : Ek^{j}_{V_m}(Ack\_Session\_Key)$

### 4.5.5 Periodic Re-keying

Periodic re-keying updates symmetric encryption keys after a certain interval. If a cryptographic key is being employed for a longer time period, the higher the chance that the key is going to be successfully cryptanalyzed. As we mentioned earlier, re-keying is not necessary for our software upload multicast session when a member joins/leaves a multicast group. However, in order to prevent any cryptanalysis attack, we recommend for periodic re-keying of multicast keys for both data and control multicast groups. The period of re-keying depends on the key size and the cryptographic algorithm used to encrypt the multicast data.

#### 4.5.5.1 Fully-trusted System

In FT-system, the re-keying of multicast SK is initiated and controlled by the group leader $B^i_{L_m}$. The $B^i_{L_m}$ generates the new SK ($k'_m$) and its expiration period, prepares *Msg_Rekeying_SK* message that consist of the message ID, new SK, its expiration period, revision number and timestamp. It delivers it to other members encrypted with $k^i_{CG_m}$ and its digital signature. Each BS verifies the message and multicasts it to underlying vehicles encrypted with the current $k_m$. Upon receiving the new SK, the $V_m$ verifies it. If it is not received successfully, it sends a NACK message to the BS encrypted with its authentication key.

$B^i_{L_m} \Rightarrow \mathbf{B}^i_m : Ek^i_{CG_m}(Msg\_\mathrm{Re}keying\_SK), Sign_{B^i_{L_m}}(H)$

$B_i \Rightarrow \mathbf{V}_{B_i} : Ek_m(Msg\_\mathrm{Re}keing\_SK), Sign_{B_i}(H)$

$V_m \to B_i : Ek_{V_m^j}(NACK\_SK)$ if new SK is not received successfully.

#### 4.5.5.2 Semi-trusted System

In ST-system, the RGM initiates and controls the re-keying process. It generates the new SK ($k'_m$) and its re-keying period, prepares *Msg_Rekeying_SK* message, encrypts with current SK ($k_m$), digitally signs it and multicasts it to the underlying vehicles via the BSs. If a vehicle cannot receive the new SK successfully, it sends the NACK as it is in FT system and the BS forwards it to the RGM.

$R_i \Rightarrow \mathbf{B}_m^i : Ek_m(Msg\_\mathrm{Re}keying\_SK), Sign_{R_i}(H)$

$B_i \Rightarrow \mathbf{V}_{B_i} : Ek_m(Msg\_\mathrm{Re}keying\_SK), Sign_{R_i}(H)$

$V_m \to B_i : Ek_{V_m^j}(NACK\_SK)$

$B_i \to R_i : Ek_{V_m^j}(NACK\_SK)$

In summary, the re-keying message can be sent as other multicast messages.

## 4.6 System Analysis

### 4.6.1 Cryptographic Overhead Analysis

The confidentiality of the messages exchanged during multicast session establishment procedure is ensured by using symmetric key encryption technique and the authenticity is guaranteed through digital signature. We assume AES (Advanced Encryption System) for symmetric key cryptography which is a symmetric-key encryption standard adopted by US Government [50] . It is a symmetric block cipher where block size is fixed to 128 bits and the key size could be 128, 192 or 256 bits. Accordingly, the AES standard comprises three block ciphers: AES-128, AES-192 and AES-256 adopted from a larger collection originally published as Rijndael**.** Here, we assume AES-128. The AES cipher is specified as a number of repetitions of transformation rounds that convert the input plaintext into the final output of ciphertext. Each round consists of several processing steps, including one that depends on the encryption key. A set of reverse rounds are applied to transform ciphertext back into the original plaintext using the same encryption key. According to [51], let $4B$, $4K$, and $R$ denote the block size (in bytes), key length (in bytes) and number of rounds of Rijndael, respectively and $T_{and}$, $T_{or}$ and $T_{shift}$ denote the numbers of processing cycles required for performing basic operations of a byte-wise AND, a byte-wise OR, and a byte-wise SHIFT, respectively. Then the total number of processing cycles (PC) to encrypt a block is given by:

$$\begin{aligned} T_E &= (16BT_{and} + 11BT_{or} + 3BT_{shift}) + \\ &[46BT_{and} + (31B+12)T_{or} + (64B+96)T_{shift}](R-1) \end{aligned} \qquad \textbf{(4.1)}$$

The total number of processing cycles to decrypt a block is given by:

$$\begin{aligned} T_D &= (16BT_{and} + 11BT_{or} + 3BT_{shift}) + [161BT_{and} + (122B+12)T_{or} \\ &+ (32B+96)T_{shift}](R-1) \end{aligned} \qquad \textbf{(4.2)}$$

Let $L$ denotes the payload size in a frame in bytes, then the number of processing cycles to encrypt and decrypt the frame are as follows, respectively,

$$O_E = \left\lceil \frac{8\times L}{4B\times 8} \right\rceil T_E = \left\lceil \frac{L}{4B} \right\rceil T_E$$

$$O_D = \left\lceil \frac{8 \times L}{4B \times 8} \right\rceil T_D = \left\lceil \frac{L}{4B} \right\rceil T_D \tag{4.3}$$

For AES-128, $B = 4$ and $R = 10$ and if we assume $T_{and} = T_{or} = T_{shift} = 1$, then $T_E = 6168$ and $T_D = 12432$ PC. Then the processing time to encrypt and decrypt a frame is $t_{PE} = T_E * \frac{1}{f}$, $t_{PD} = T_D * \frac{1}{f}$ where $f$ is the speed of the processor.

For digital signature generation and verification, we assume DSA (Digital Signature Algorithm) with key 1024-bit key length and SHA-1 (Secure Hash Algorithm) which is used in DSS (Digital Signature Standard) [52]. The DSA is used to provide data integrity, authenticity and non-repudiation. In [53], the performance of popular public-key encryption algorithms and hash functions, such as DSA and Data Encryption Standard (DES) was studied. They showed that on an 8-bit commercial microcontrollers running at 10 MHz, the processing time of signing with DSA is 100 ms and verifying the signature is 160 ms. Now-a-days, vehicle's ECUs usually use 16- and 32-bit microcontroller that run at 66~132 MHZ. Current advancement in vehicular technology may require for higher speed processor. In this paper, we assume microcontroller with speed 500 MHz for vehicle's ECUs and 1 GHz processor for BSs and RGMs.

### 4.6.2 Receiving Buffer Delay/Queue Delay Model

The CM, RGMs and BSs maintain a FIFO buffer for message reception and processing. Queue delay is the time that a message has to wait in the buffer of an RGM or a BS for service. The service time is mainly the time to send a message either through the wired or the wireless channel. Since wired channel is assumed noise free and bandwidth is fixed, the message transmission delay can be assumed to be "Deterministic". On the other hand, the wireless channel is noisy due to path loss and the bandwidth is limited and variable. Hence, the message transmission delay can be modeled as "General" [19]. The buffer size is assumed to be high so that no message is dropped while receiving. Thus, the queue delay of entities that send message through wired and wireless channel can be modeled as an $M/D/1$ and $M/G/1$ queue, respectively. For $M/G/1$ queue, the average waiting time in the queue, $E(W)$ is given by [54]

$$t_{RB} = E[W] = \frac{\lambda E[{T_D}^2]}{2(1-\rho)} \qquad \textbf{(4.4)}$$

where $E[T_D]$ is the mean processing time of a message, $\rho = \lambda E[T_D]$ is the server load and $\lambda$ is the message arrival rate for Poisson arrival process.

For deterministic service time, $E(W)$ is given by,

$$t_{RB} = E[W] = \frac{\rho E[T_D]}{2(1-\rho)} \qquad \textbf{(4.5)}$$

### 4.6.3 Transmit Buffer Delay

The transmit buffer delay is the time that elapse between the time when a message is ready to send and access the channel using CSMA/CD or CSMA/CA channel access mechanism in wired or wireless channel, respectively. In wired part of the network, the delay model for 1-persistent CSMA/CD is used which is described in Section 2.1. In the wireless part, the delay model for CSMA/CA with BEB is used which is described on Section 2.2.

# 5 CHAPTER 5 – RESULTS AND PERFORMANCE ANALYSIS

## 5.1 Simulation Model

For unicast software upload, we generated a uniformly distributed random number using drand48() function in C++ with gcc compiler for a particular packet error probability *p* due to hacking. If the random number is less than *p* then the packet was considered as a bad packet and vice versa.

In order to simulate wireless multicasting, we developed an integrated Vehicular Wireless Communication Network (VNET) simulator which combines a realistic traffic mobility model and a wireless communication network model based on DEVS (Discrete Event System Specification) [55]. The software tool "DEVS#" [56] is used for modeling and simulation of the proposed multicast architecture. It is an object-oriented implementation of the DEVS formalism written in C# language. The details of the DEVS theory and VSDN simulator is presented in Appendix A.

We simulate 100 miles of an uninterrupted four lanes freeway with macroscopic traffic flow model where vehicles move at a speed determined by the speed limit of the road. Four levels of constant speeds, 75, 70, 65 and 60 mph, are taken into account, which can be considered as speed limit in each different lane. Vehicles are generated from a Poisson process with a parameter λ veh/hr. A vehicle entering under a BS's cell computes its speed using a truncated version of normal distribution with mean as the speed limit of the road and standard deviation 10% of the speed limit [57]. The range of each BS is assumed to be 2 miles. For communication network simulator, we simulated MAC and physical layers of both wired and wireless communication channels. In the wired part of the network we implemented CSMA/CD and in the wireless part, CSMA/CA channel access mechanisms. For receiving part of wired channel, we assumed noiseless channel, hence all successfully transmitted packets are guaranteed to be received by the destination. On the other hand, the wireless channel is assumed noisy and simulation results are collected for varying Packet Error Rate (*PER*). The values of other parameters used in the simulation are listed in Table 5.1.

TABLE 5.1 VARIOUS PARAMETERS VALUES USED IN SIMULATION.

| **Parameter** | **Value** | **Parameter** | **Value** |
|---|---|---|---|
| Range of BS, $R$ | 2 mile | Wireless Control rate | 1 Mbps |
| Wired Bus speed, $C$ | 10 Mbps | PHY header | 192 bits |
| Slot time for wired channel, $\sigma$ | 51.2 μS | MAC Header | 224 bits |
| Propagation Delay for wired channel, $\delta$ | 26 μS | ACK packet | 112 bits + PHY |
| Jam time after collision, $t_J$ | 3.2 μS | Initial Backoff Window size | 32 |
| Retry Limit for wired channel, $M$ | 16 | Maximum backoff stage | 5 |
| Wireless Channel Bit Rate, $R$ | 10 Mbps | Wireless Retry Limit | 7 |
| Slot time for wireless channel, $\sigma_w$ | 20 μS | DIFS | 50 μS |
| Wireless propagation delay | 1 μS | SIFS | 10 μS |

## 5.2 Performance Analysis of Unicast Software Upload

Figure 5.1 and Table 5.2 show the resemblance between the analytical and simulation results for the average number of packet transmissions for the Single-copy and

Multiple-copy software upload techniques, respectively. For the Single-copy transmission, at higher $p$ the average numbers of packet transmissions ($N$) for successful software upload increases exponentially as the software size increases. However, if the software is sent in segmented form, it reduces $N$ considerably. Figure 5.2 exemplifies the effect of segmentation for the software size with 1024 packets and different number of segments. The more the number of segments, the lesser is the number of packet transmissions necessary for successful software upload. Conversely, as the number of segments increases, it might take more time to encrypt, decrypt and transmit all the segments. Hence, there should be a trade-off between number of segments and processing time.

The two-copy software upload is always superior to the Single-copy software upload as long as security is concerned. Since the second copy will be transmitted after a random time interval in a random packet order, it is very unlikely that an intruder would know whether a second copy will be transmitted or not. Moreover, even if an intruder changes one packet of the first copy, it would be difficult for him to change the same packet in the second copy due to the randomness of packet transmission. Figure 5.3 shows the average number of packet transmissions ( $N_p$ ) to upload a single packet successfully in the multiple-copy software upload scenario. Unlike the single-copy

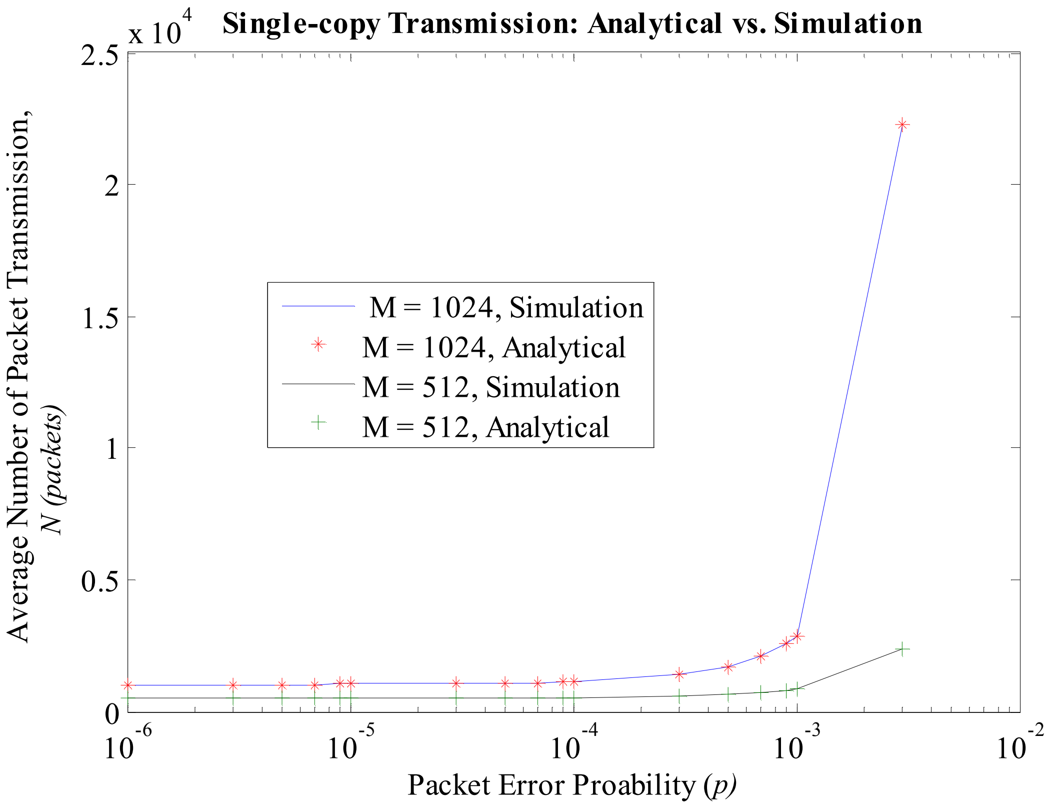


**Figure 5.1 Comparison of Analytical and Simulation Results for Single-copy Software Upload Technique.**

software upload, the total number of packet transmissions necessary to upload the entire software is linearly dependent on the software size (eq. (3.8), (3.11) and (3.13)). For lower values of *p,* on an average only two packets need to be transmitted for any of the techniques mentioned here. For higher values of *p*, *Finite Buffer with Random Packet Delete* requires the least number of packet transmissions with respect to the ideal case where we have infinite number of buffers. In general, the hacking probability is very low. Thus, any of the techniques could be used if there are one or more unmatched packet pairs. In addition, *N* does not vary notably between the two-buffer case and infinite-

TABLE 5.2 COMPARISON OF ANLYTICAL AND SIMULATION RESULTS OF DOUBLE COPY SOFTWARE TRANSMISSION.

| ***p*** | **Infinite Buffer** | | **Finite Buffer with Pair Transmission** | | **Finite Buffer with Two Consecutive Good Packets** | | **Finite Buffer with Random Packet Delete** |
|---|---|---|---|---|---|---|---|
| | **$N_p$ (Sim.)** | **$N_p$ (Analytical)** | **$N_p$ (Sim.)** | **$N_p$ (Analytical)** | **$N_p$ (Sim.)** | **$N_p$ (Analytical)** | **$N_p$ (Sim.)** |
| 0.1 | 2.2228 | 2.2222 | 2.4704 | 2.4691 | 2.3443 | 2.3457 | 2.2847 |
| 0.01 | 2.0204 | 2.0202 | 2.0406 | 2.0406 | 2.0303 | 2.0304 | 2.0253 |
| 0.001 | 2.0020 | 2.0020 | 2.0040 | 2.0040 | 2.0031 | 2.0030 | 2.0024 |
| 0.0001 | 2.0002 | 2.0002 | 2.0004 | 2.0004 | 2.0003 | 2.0003 | 2.0003 |
| 0.00001 | 2.0000 | 2.0000 | 2.0000 | 2.0000 | 2.0000 | 2.0000 | 2.0000 |

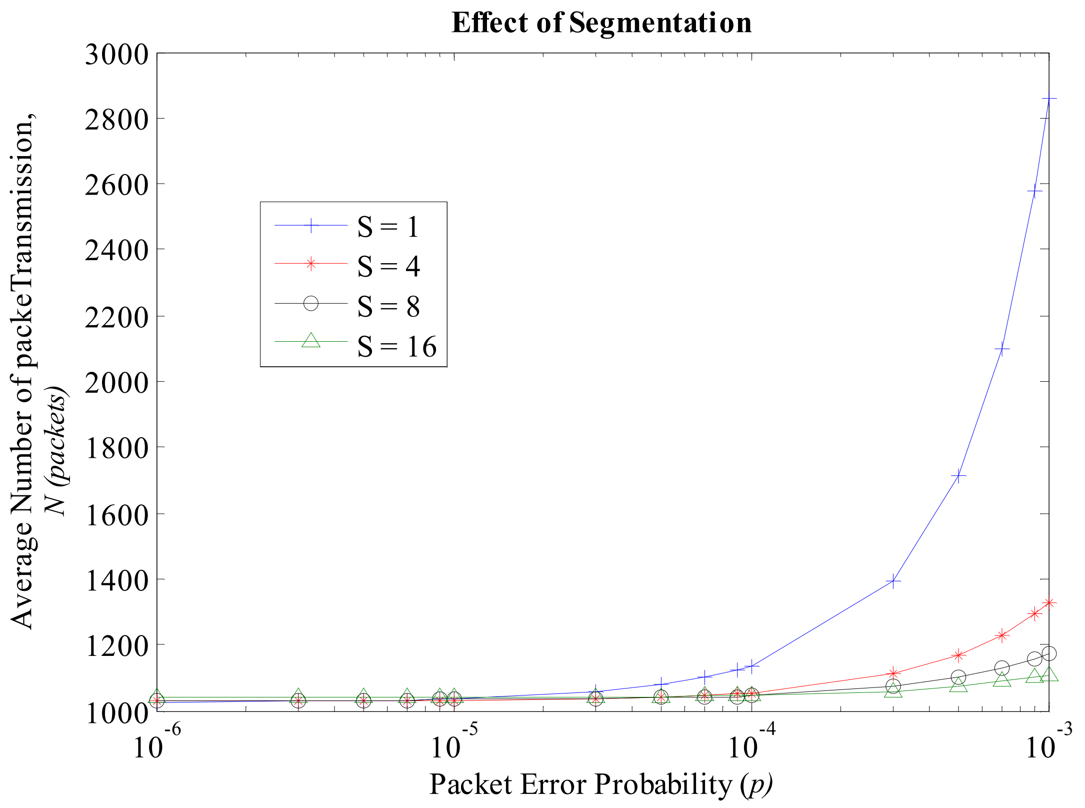


**Figure 5.2 Effect of segmentation on Single-copy Software Upload for M = 1024.**

buffer case. Addition of more buffers would not increase the performance of software upload remarkably. Consequently, we propose to use not more than two buffers in vehicle's software modules to upload two copies of software.

At lower $p$, single-copy software upload requires fewer number of packet transmission than the multiple-copy software upload. However, the later technique offers additional security if the software packets are transmitted in random order and the second copy is transmitted after a random time-interval with a long average value.

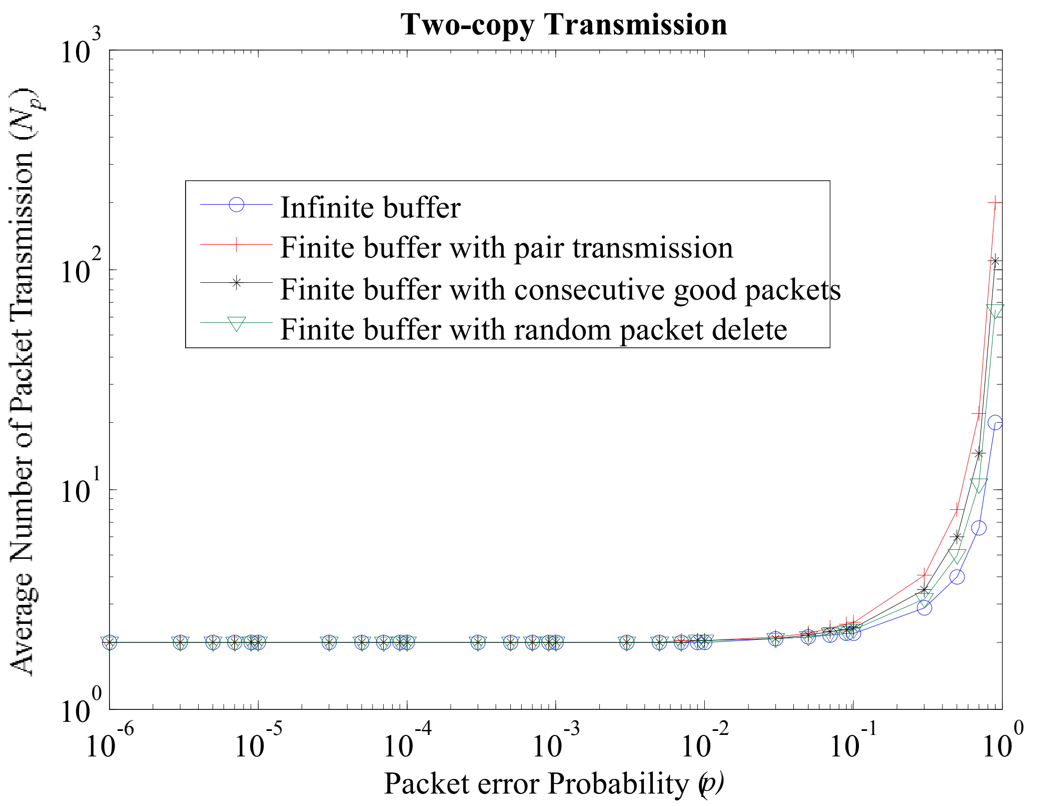


**Figure 5.3 Average number of packet transmission ( ) for successful upload of a single packet for Two-copy software Upload.**

## 5.3 Performance Analysis of Multicast Software Upload

In measuring the efficiency of the proposed GKM protocols, we consider the criteria referred in Section 2.7. We find the communication overhead, computation overhead and storage overhead of multicast session initialization and handoff procedures, for both the FT and ST systems. We also show the performance of the proposed protocols in terms of scalability, bandwidth requirements, strength against security attacks and cost requirements. Since re-keying of SK can be considered as an additional multicast data, the overhead for this procedure has not been evaluated.

### 5.3.1 Communication Overhead

In finding the communication delay, we add up the Round Trip Delay, $T_{RTD}$ of each message to be successfully transmitted between two nodes, used in the protocol. The $T_{RTD}$ consists of the transmission delay $t_D$, receiving buffer delay ( $t_{RB}$ ) and processing delay ( $t_P$ ) at destination node. In Figure 5.4, the delay performance of 1-persistent CSMA/CD is plotted for $a = 0.44$. As the offered load increases, the average time to transmit a packet increases. Note that, average packet transmission time for wired part of the network is in the range of ms.

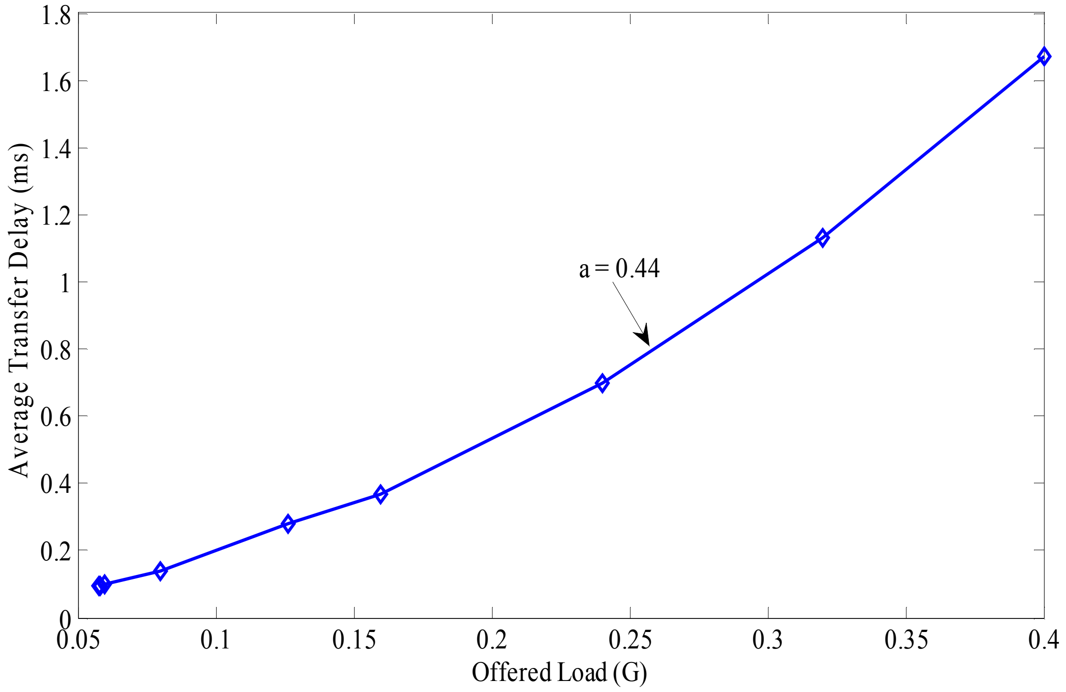


**Figure 5.4 Average Message Transmission Delay of Wire Network for Normalized Propagation Delay a = 0.44.**

From the wireless message transmission delay model it is seen that the distribution of message transmission delay mainly depends on the pseudo collision probability $p$ whereas the maximum value of $p$ is determined by the number of active stations, $n$. Table 5.3 shows maximum value of $p$ for different number of stations and different *PER*. Figure 5.5 shows the average packet transmission delay $E[T_D]$ as a function of $n$ for different *PER*. From Figure 5.5 it is seen that for lower values of *PER*, $E[T_D]$ hardly depends on *PER*. However, for higher values of *PER*, $E[T_D]$ increases with *PER*. Packet arrival rate at each station is adopted from simulation and the receiving

TABLE 5.3 SATURATION VALUE OF COLLISION PROBABILITY

| $n$ | Max $p$ (PER = 0.0001) | Max $p$ (PER = 0.0025) | Max (PER = 0.001) $p$ | Max $p$ (PER = 0.01) | Max $p$ (PER = 0.05) |
|---|---|---|---|---|---|
| 10 | 0.2899 | 0.291 | 0.2903 | 0.2942 | 0.3116 |
| 15 | 0.3555 | 0.3558 | 0.3553 | 0.3584 | 0.3722 |
| 20 | 0.4 | 0.4007 | 0.4003 | 0.4029 | 0.4148 |
| 25 | 0.43429 | 0.4349 | 0.4345 | 0.4368 | 0.4475 |

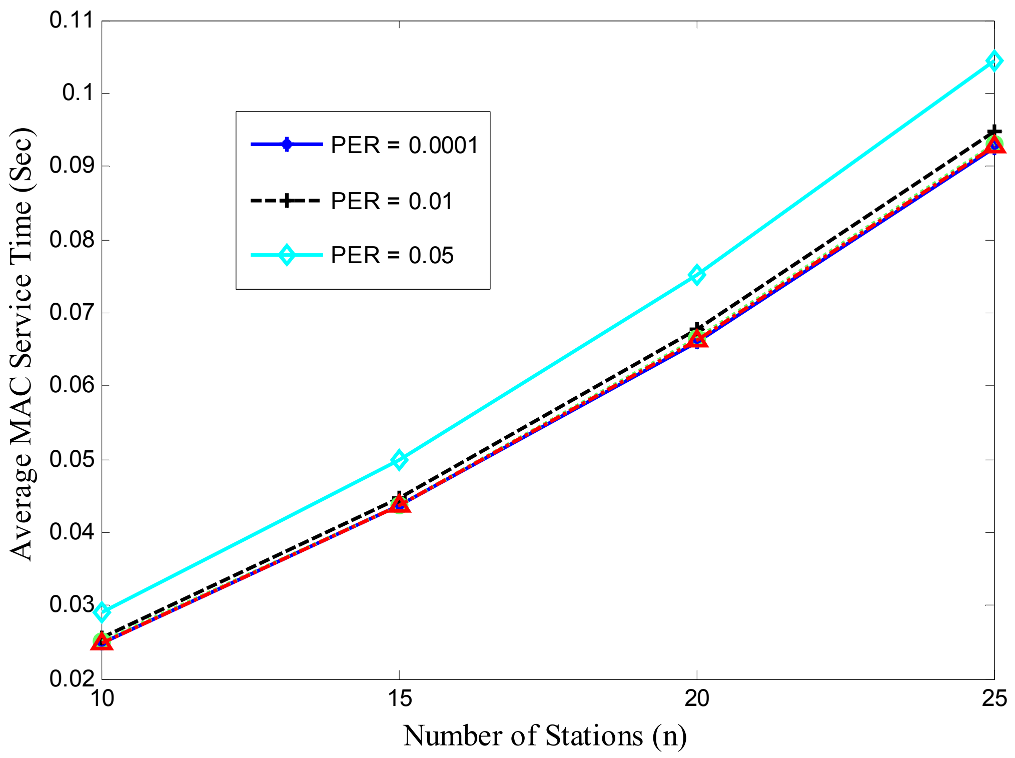


**Figure 5.5 Average message transmission delay of CSMA/CA protocol**.

buffer delay is calculated using equations (4.4) and (4.5) for wireless and wired channel, respectively.

In measuring the performance of proposed GKM protocols, we calculate the *Multicast Session Initialization Latency* ($L_I$) and *Hand-off Latency* ($L_H$) of the proposed GKM protocols. The $L_I$ is defined as the average time elapse between the submission of an *Ack_DG_Join* message by a vehicle and the reception of *Ack_Session_Key* message by the BS (in the FT system) or RGM (in the ST system). The $L_H$ is defined as the average time elapsed between the submission of *Req_Handoff_Joining* message by the vehicle and the reception of *Ack_Session_Key* message by the BS or RGM depending on FT or ST system, respectively.

Let,

$T_I$ = Multicast initialization latency per vehicle in the absence of any handoff.

$N_B, N_M, N_R$ = Number of *BS*-, *MG*- and *RGM-level* handoff per vehicle during session initialization, respectively.

$T_B, T_M, T_R$ = Additional delay incurred for *BS*-, *MG*- and *RGM-level* handoff, respectively. Then,

$$L_I = T_I + N_B * T_B + N_M * T_M + N_R * T_R$$

The values of $N_B, N_M, N_R$ are adopted from simulation. In measuring $L_I$ , we investigated the effect of the *CG-size* (no. of BSs/CG), vehicle arrival rate ($\lambda_i$ veh/hr/lane) and wireless *PER*. In general, the ST system takes few ms higher times than the FT system and initialization latency increases considerably with the number of handoff that occurs during session establishment. From the multicast session initialization technique described above, it is noted that both the FT and ST systems require same number of wireless messages whereas the ST system requires more number of wired messages. However, wired message transmission delay is insignificant compared to the wireless message transmission delay. Handoff during session initialization requires vehicle's authentication key to be transferred to the joining BS/RGM (FT/ST system), the joining

BS/RGM needs to send its public key to the vehicle and the vehicle needs to be sent the SK of new MG. These additional messages required to adopt handoff increases the $L_I$.

TABLE 5.4 EFFECT OF *CG-SIZE* ON MULTICAST INITIALIZATION LATENCY ($L_I$)
($PER = 0.0001$, $\lambda_i = 200\ veh/hr/\ln$, NO. OF RGM = 2).

| CG-Size | Fully-Trusted System | | Semi-Trusted System | |
|---|---|---|---|---|
| | $L_I$[95% CI] (Sec) (Simulation) | $L_I$ (Sec) (Analytical) | $L_I$[95% CI] (Sec) (Simulation) | $L_I$ (Sec) (Analytical) |
| 1 | 2.3374[2.2609,2.4141] | 2.3606 | 2.4956[2.3438,2.6474] | 2.4228 |
| 3 | 2.3964[2.2506,2.5422] | 2.3705 | 2.4384[2.3494,2.5274] | 2.4077 |
| 4 | 2.3318[2.2614,2.4021] | 2.3412 | 2.4223[2.2558,2.6689] | 2.4114 |
| 5 | 2.3421[2.2837,2.4003] | 2.3601 | 2.4606[2.4174,2.5037] | 2.419 |

The number of BSs in a CG (i.e., *CG-size*) determine the number of MG under an RGM, i.e., $N_M^i = \frac{N_B^i}{N_B}$. As the *CG-size* ($N_B$) decreases, the number of MG, $N_M^i$ increases. Thus, the *MG-level* handoff increases. However, multicast session initialization technique handles both the *BS*- and *MG-level* handoff in a similar fashion. In both cases, the leaving BS needs to send the vehicle authentication key to the joining BS and the joining BS authenticate the vehicle and send the SK. Therefore, the *CG-size* does not have any significant effect on $L_I$. Table 5.4 shows the effect of *CG-size* on $L_I$ for both FT and ST system. However, during multicast session if the *CG-size* is larger the vehicle does not need to get the new SK when moving from one BS to another unless the joining BS is under a different MG. This will reduce handoff latency during multicast session.

TABLE 5.5 EFFECT OF WIRELESS PER ON MULTICAST SESSION INITIALIZATION LATENCY($L_I$)
(CG-SIZE = 4 BSS/MG, $\lambda_i = 200\ veh/hr/\ln$, NO. OF RGM = 2).

| **PER** | **Fully-trusted System** | | **Semi-Trusted System** | |
|---|---|---|---|---|
| | $L_I$[95% CI] (Sec ) (Simulation) | $L_I$ (Sec) (Analytical) | $L_I$[95% CI] (Sec ) (Simulation) | $L_I$ (Sec) (Analytical) |
| 0.0001 | 2.3318[2.2614,2.4021] | 2.3606 | 2.4223[2.2558,2.6689] | 2.4114 |
| 0.001 | 2.3496[2.2248,2.4744] | 2.3677 | 2.4392[2.3204,2.558] | 2.4283 |
| 0.0025 | 2.3896[2.2791,2.5] | 2.3826 | 2.4623[2.4396,2.485] | 2.449 |
| 0.01 | 2.4316[2.1819,2.6814] | 2.4435 | 2.5385[2.4408,3.0724] | 2.5136 |
| 0.05 | 2.812[2.5109,3.113] | 2.8417 | 2.9479[2.8234,3.0724] | 2.9356 |

Table 5.5 shows the effect of wireless *PER* on $L_I$ for both the FT and ST systems. The higher the *PER*, the number of retransmissions required to send a message successfully to the receiver will increase. This increases the average message transmission delay in the wireless part of the network. Again, increase in message transmission delay cause more handoff to happen during session establishment since a vehicle need to wait longer under a BS before being authenticated and receive SK. Hence, the $L_I$ increases with higher *PER* as it is seen in Table 5.5.

TABLE 5.6 EFFECT OF VEHICLE ARRIVAL RATE ON MULTICAST SESSION INITIALIZATION LATENCY ($L_I$)
(CG-SIZE = 4 BSS/MG, *PER* = 0.0001, NO. OF RGM = 2).

| $\lambda_i$ veh/hr/ln | **Fully-trusted System** | | **Semi-Trusted System** | |
|---|---|---|---|---|
| | $L_I$[95% CI] (Sec ) (Simulation) | $L_I$ (Sec) (Analytical) | $L_I$[95% CI] (Sec ) (Simulation) | $L_I$ (Sec) (Analytical) |
| 100 | 1.0337[0.9581,1.1094] | 1.005 | 1.8503[1.6957,2.0049] | 1.821 |
| 200 | 2.3318[2.2614,2.4021] | 2.3606 | 2.4223[2.2558,2.6689] | 2.4114 |
| 300 | 3.2063[3.1017,3.3109] | 3.2632 | 3.3245[3.1529,3.4961] | 3.3441 |

The effect of $\lambda_i$ on $L_I$ is presented on Table 5.6. Since $\lambda_i$ is independent and identically distributed, the total arrival rate of vehicles is $\lambda = \sum_{i=1}^{L_N} \lambda_i$ veh/hr, where $L_N$ is the total number of lanes in the road. As $\lambda_i$ increases, the number of vehicles under each BS increases which increases the message arrival rate at BS, RGM and CM. Consequently, message transmission delay and receiving buffer delay in both wired and wireless part increases. Hence, $L_I$ increases with $\lambda_i$. However, since the road trajectory is known, there is an upper limit of the number of vehicles that could reside under a BS at a certain time. That makes the architecture practicable.

TABLE 5.7 EFFECT OF NUMBER OF RGM ON MULTICAST SESSION INITIALIZATION LATENCY ($L_I$)
($CG\text{-}SIZE = 4$ BSs/MG, $PER = 0.0001$, $\lambda_i = 200\ veh/hr/\ln$).

| No. of RGM | Fully-trusted System | | Semi-Trusted System | |
|---|---|---|---|---|
| | $L_I$[95% CI] (Sec) (Simulation) | $L_I$ (Sec) (Analytical) | $L_I$[95% CI] (Sec) (Simulation) | $L_I$ (Sec) (Analytical) |
| 1 | 2.3224[2.1647,2.48] | 2.3677 | 2.3954[2.378,2.4128] | 2.4077 |
| 2 | 2.3318[2.2614,2.40214] | 2.3606 | 2.4223[2.378,2.4128] | 2.4114 |
| 4 | 2.3426[2.1104,2.5749] | 2.3677 | 2.4204[2.3731,2.4676] | 2.4171 |

The Auto Company or the Software Vendor may decide on the number of Regional Offices in certain coverage area. From GKM perspective, when there are more regional offices in a certain area, *the RGM-level* handoff increases. The *RGM-level* handoff requires handoff notification and handoff joining messages to be forwarded to the corresponding RGMs. Upon receiving the handoff leaving message, the leaving RGM sends the vehicle authentication key to the joining RGM. The joining RGM sends the vehicle authentication key to the joining BS and the BS sends its public key and multicast session key to the vehicle in case of the FT system or the joining RGM sends its public key and multicast SK to the vehicle via the joining BS in case of the ST system. In summary, *RGM-level* handoff increases the initialization latency $L_I$ in both the FT and ST systems. Table 5.7 shows the effect the number of RGM on $L_I$ which is simulated over 100 miles area. Moreover, the more Regional offices, the higher operational cost it would be for the AC or SV. However, if there is less number of RGMs, there is possibility of single-point failure.

Table 5.8 shows the analytical result of $L_H$ to perform each type of handoff during multicast session. Note that, the *BS*- and *MG-level* handoff latency for ST system is lower than the FT system. In the FT system, both the *BS*- and *MG-level* systems require the vehicle authentication key to be transferred to the joining BS, and the joining BS needs to send its public key to the vehicle. On the other hand, in ST system RGM does not need to send the vehicle authentication key and its public key since the vehicle is still under the same RGM. However, *RGM-level handoff* in ST system requires some additional wired messages to be transmitted which cause $L_H$ to be little higher in ST system than the FT system.

TABLE 5.8 COMAPRISON OF HANDOFF LATENCY ($L_H$)
(PER = 0.0001, CG-SIZE = 4, BSS/CG, NO. OF RGM =2, $\lambda_i = 200\ veh/hr/\ln$).

| $\lambda_i$ **veh/hr/ln** | $L_H$ **(Sec) (Fully-Trusted System)** | | | $L_H$ **(Semi-Trusted System)** | | |
|---|---|---|---|---|---|---|
| | *BS-level* Handoff | *MG-level* Handoff | *RGM-level* Handoff | *BS-level* Handoff | *MG-level* Handoff | *RGM-level* Handoff |
| 100 | 1.4562 | 1.8999 | 1.9001 | 1.3849 | 1.4132 | 1.9019 |
| 200 | 2.2728 | 2.3648 | 2.3826 | 2.2559 | 2.2559 | 2.4524 |
| 300 | 4.0179 | 4.2235 | 4.2242 | 3.7736 | 3.7736 | 4.3066 |

### 5.3.2 Storage Overhead

In calculating the storage overhead of proposed key management protocols, we determine the number of keys that each entity involved in GKM needs to maintain. We classify the keys in three categories: pair-wise shared key, group-wise shared key and public/private key pair for asymmetric encryption. The arrangements of these three types of keys from the point of view of the entities are summarized in Table 5.9.

TABLE 5.9 ARRANGEMENTS OF KEYS OF THE ENTITIES INVOLVED IN GKM.

| Entity | Pair-wise Shared Key | Group-wise Shared Key | Public/Private Key |
|---|---|---|---|
| **Vehicle** (**$V_m$**) | A set of Authentication Keys ($\mathbf{k}_{V_m}(k^1_{V_m},\cdots,k^n_{V_m})$) shared between $V_m$ and the RGM. | Multicast session key ( $k_{C_m}$ ) shared by the multicast group members. | 1. Public key of the RGM to which it is registered.<br>2. Public key of the BS under which it is currently residing if the system is fully-trusted. |
| **Base Station** (**$B_i$**) | 1. BS authentication key ($k^i_{B_m}$) shared between a BS $B^i_m$ and the RGM $R_i$.<br>2. BS-Leader private key ($k_{LB_m}$), shared between the group leader $B^i_{L_m}$ and a BS $B^i_j$ for FT-system. | 1. All-BS session key ($k_{A\_BS}$) for FT-system which is used by the RGM to multicast the software packets to its underlying BSs. This short term key is shared between an RGM and a set of BSs residing under that RGM.<br>2. Control- Group key ( $k^i_{CG_m}$ ) shared among the BSs $\mathbf{B}^i_m$ in a control Group.<br>3. Multicast session key | 1. Its own Public/Private key pair.<br>2. Public key of the RGM under which it resides.<br>3. Public keys of neighboring BSs under the RGM ($P_A * N^i_B$). |

| | | | |
|---|---|---|---|
| | | ($k_{C_m}$) if the infrastructure is fully-trusted. | |
| **RGM** **($R_i$)** | 1. RGM authentication key ($k_{R_i}$) shared between the AC and an RGM<br>2. BSs authentication keys ($N_B^i * k_{B_i}$)<br>3. Authentication keys of vehicles registered under the RGM. ($N_V^i * \mathbf{k}_{V_m}$) | 1. All-BS session key ($k_{A\_BS}$) for fully-trusted system.<br>2. Multicast session keys ($N_C * k_{C_i}$) if the infrastructure is semi-trusted. | 1. Its own Public/Private key pair.<br>2. Public key of the BSs reside in that region. |

In both the FT and ST systems, the RGM stores authentication keys of vehicles that are registered under its region. In the ST system, the RGM generates and distributes session keys of all MGs under its region. Hence, it has to store more keys compared to the RGM in FT-system. On the other hand, a BS in the FT-system has to store a CG key, multicast SK and vehicles' authentication keys for that software update session. Hence, BSs in FT-system require maintaining more keys than the BSs in ST-system. For FT-system the vehicle needs to store the BS's public key in addition to its authentication key, multicast SK and the RGM's public key.

### 5.3.3 Computational Overhead

Various computational overhead performed by each entity during multicast session initialization, handoff and re-keying are shown in Table 5.10 for both the FT and ST systems. The notation $E$ and $D$ stand for encryption and decryption operations, $PuKey$ is the public key operation and $\Pr Key$ is the private key operation, $N_{RH}$, $N_{MH}$ and

$N_{BH}$ are total number of *RGM-level*, *MG-level* and *BS-level* handoff, respectively. For FT-system, the RGM delegates most of the computations to the BSs whereas in a ST-system the BSs act as intermediate entities to relay the messages from the RGM to the vehicles. Hence, they need to perform very little computation in comparison to the RGM. The vehicles need to perform same amount of computation for both the FT and ST systems.

TABLE 5.10 COMPARISON OF COMPUTATIONAL OVERHEAD FOR FT AND ST SYSTEMS.

| | **Entity** | **Fully-trusted System** | **Semi-trusted System** |
|---|---|---|---|
| **Multicast Session Initialization** | *RGM* | $N_{RH}(\Pr Key + PuKey) + N_V E$ | $\Pr Key + N_V(E+D) + N_{RH}(\Pr Key + PuKey)$ |
| | $B_i$ | $\Pr Key + N_{VB_i} * (E+D) + (N_{MH} + N_{BH}) * (\Pr Key + PubKey)$ | -- |
| | $V_m$ | $PuKey + E + D$ | $PuKey + E + D$ |
| **BS- and MG-level hand-off** | $B_1$ | $PuKey + \Pr Key$ | -- |
| | $B_2$ | $PuKey + \Pr Key + E + D$ | -- |
| | $V_m$ | $2E + D$ | $2E + D$ |
| | *RGM* | -- | $E + D$ |
| **RGM-level** | $B_1$ | $E$ | -- |
| | $B_2$ | $2E + D$ | -- |

| | | | |
|---|---|---|---|
| | $V_m$ | $2E + D$ | $2E + D$ |
| | $RGM1$ | $D + PuKey + \Pr Key$ | $D + PuKey + \Pr Key$ |
| | $RGM2$ | $PuKey + \Pr Key + E$ | $PuKey + \Pr Key + E + D$ |

### 5.3.4 Security Analysis

In the proposed GKM protocols, each entity involved in software distribution requires mutual authentication before start any communication. Since the CM, RGM and BS are fixed entities, authentication is done using a standard mutual authentication protocol such as Internet Protocol Security (IPSec), Secure Socket Layer (SSL), Transport Layer Security (TLS), Secure Shell (SSH) or authenticated Diffie-Helman method. We assume that each vehicle will equipped with a set of authentication keys. One key is used only in one session to authenticate the vehicle. The authentication key is shared only with the RGM and/or trusted BSs. Since the multicast SK is sent to each vehicle individually encrypted with its authentication key, any unauthorized vehicle cannot obtain the SK. This provides authentication as well as confidentiality of the SK. During multicast session, any handoff message sent by the vehicle is encrypted with its authentication key which provides non-repudiation of the message. When an RGM or a BS sends authentication key of a vehicle to another RGM or BS, or sends any re-keying or data request message, it digitally signs the message to provide authenticity, integrity and non-repudiation. Moreover, each message will have timestamp that prevents replay attack. It is important to protect vehicles' ECUs and memory buffers that contain proprietary software from both outside and inside attackers, for example, unauthorized employee or ex-employee. We propose to design vehicles' ECUs and memory buffers as read protected and tamper resistant devices so that no one can retrieve proprietary information by implementing any kind of security attack.

# 6 CHAPTER 6 – CONCLUSION

This thesis presents detail architecture of RSU in advance vehicles' software modules using existing wireless communication technologies such as Wi-Max, Wi-Fi or cellular. In order to upload software in a single vehicle, wireless unicasting is proposed where the BSs act as proxies to reliably and honestly relaying the software packets from the SV to the vehicle. Since they do not have access to the software packets, it eliminates any security threat that might exist if the BSs locally decrypt and encrypt the packets. The architecture provides mutual authentication of the SV and the vehicle. A vehicle's authentication keys are shared between the AC and the vehicle, and different authentication keys are used for different software distribution sessions which prevent known-key attack. Through analysis and simulation it is shown that if two copies of the software is sent to the vehicle, the security level increases considerably. Moreover, digital signature of the SV ensures non-repudiation and the MD of the entire software provides integrity of the software.

If the AC needs to upload software to a large number of vehicles, then wireless multicasting would be a better solution than multiple unicasting to individual vehicles. In this thesis, infrastructure based wireless multicasting is proposed where software packets are first routed to the BSs and then transmitted to the desired vehicles. Two GKM protocols based on the trust level on BSs are proposed. The RSU technique requires putting some level of trust on the BSs. If the BSs can be partially trusted, more secured system can be built. Analytical and simulation results showed that with partially trusted

BSs, it is possible to achieve almost similar performance as it is for fully-trusted BSs. The practicality of this protocol is demonstrated by simulating a VSDN which consists of a realistic traffic model and wireless communication model. For a BS range of 2 miles and a vehicle speed of 60-70 mph, a vehicle resides under a BS for about $1\frac{1}{2}$ to 2 minutes. However, authentication and SK transport takes only few seconds. The rest of the time a vehicle can receive the actual software packets. The AC could use the ITS infrastructure or any cellular infrastructure to remotely upload software in vehicles' electronic modules. This eliminates the wireless infrastructure building cost for Auto Companies. Moreover, software upload in vehicles' ECUs is not a real-time process like inter-vehicle communication for pre-crash warning that requires high bandwidth. The software packets do not need to be transmitted within a short period of time and it is not a real-time application. Hence, the bandwidth requirement of the wireless link for uploading software is not a vital issue. To avoid any bandwidth limitations for the overall system, which includes the cellular or ITS infrastructure, software-updating process can be done during off-peak hours. Hence, it would not require lots of additional money; rather, if implemented successfully, multicasting in vehicular network can support not only the remote software upload, but also other numerous ITS applications such as distribution of weather information, optimum route information, traffic information to drivers and so on. In our future work, we plan to investigate the effect of various network component failures such as BSs or RGM. The protocol should be robust against failure to various network components such as BS or RGM.

# APPENDIX A

## Introduction to DEVS

The DEVS is a modular and hierarchical formalism for modeling and analysis of Discrete Event Systems (DESs) [55]. DEVS represents a complex model as a composite of basic models integrated hierarchically using input/output ports and couplings. The advantages of such modular constructions are [55]:

1. The DEVS framework supports scalability and reusability through the use of one model as a basic component of another model.
2. Each model in a model base system can be independently tested by coupling a test module to it. This allows verifying large complex simulation models in an incremental fashion.
3. DEVS framework supports parallel and distributed simulation of models. Hence, it is possible to develop and deploy very large-scale complex models.
4. It supports discrete event approximation of continuous systems.
5. Object oriented implementation of DEVS formalism is possible.

The DEVS formalism defines two types of models: *atomic* and *coupled*. An atomic model is one that cannot be decomposed further while a coupled model can be decomposed into component models. The atomic model defines the system behavior whereas the coupled model describes the system structure. Formally, an atomic DEVS model is defined by a 7-tuple structure:

$A =< X, Y, S, \delta_{\text{int}}, \delta_{ext}, \lambda, \tau >,$

where $X$ is a set of input events, $Y$ is a set of output events, $S$ is a set of states, $\delta_{int}$ is the internal transition function, $\delta_{ext}$ is the external transition function, $\lambda$ is the output transition function, $\tau : \mathrm{T}^{+}_{0,\infty}$ is the time advance function.

A DEVS coupled model is defined as:

$N =< X, Y, D, \{M_i\}, EIC, EOC, IC, SELECT >,$

where $D$ is the names of sub-component set, $\{M_i\}(i \in D)$ is the DEVS models (atomic or coupled) set interacting through their interface, $EIC \subseteq X \times \bigcup_{i \in D} X_i$ is a set of external input couplings, $EOC \subseteq \bigcup_{i \in D} Y_i \times Y$ is a set of external output couplings, $IC \subseteq \bigcup_{i \in D} Y_i \times \bigcup_{i \in D} X_i$ , $SELECT$ is the tie breaking selector. When the coupled model $N$ receives an input event, the coupled DEVS transmits the input event to the sub-components through *EOC*. When the sub-component produces its output event, the coupled DEVS transmit it to the other sub-components through *IC*. It also produces output event for $N$ through EOC. For an example, in VNET, many vehicles share a communication channel to send various messages to other vehicles or BS. Figure 7.1 (a) shows an example of coupling relation between DEVS models that consist of an atomic model W_Net (wireless channel) and a coupled model V1 (vehicle 1). The V1 consists of two atomic models – V1_Main which is the main part of a vehicle and V1_TX which performs as a transceiver. There are other vehicles connected to the W_Net in similar fashion. The V1_Main first sends a message through the output port 'oCom' to V1_TX, V1_TX transmits the message to the W_Net via its output port 'oComV' which is coupled with the output port 'oComV' of V1. The

W_Net forwards the message to destination through one of its output port coupled with the input port of the destination vehicle. Similarly, if V1 is the destination for any message then W_Net transmits it to V1_TX through output port 'V1' which is coupled with the input port 'iComV' of V1. Since input port 'iComV' of V1 is coupled with the input port 'iComV' of V1_TX, V1_TX receives the message and forwards it to V1_Main through output port 'oCom'. Each atomic model has its own state transition diagram. Figure 7.1(b) represents the hierarchical DEVS construction of this system.

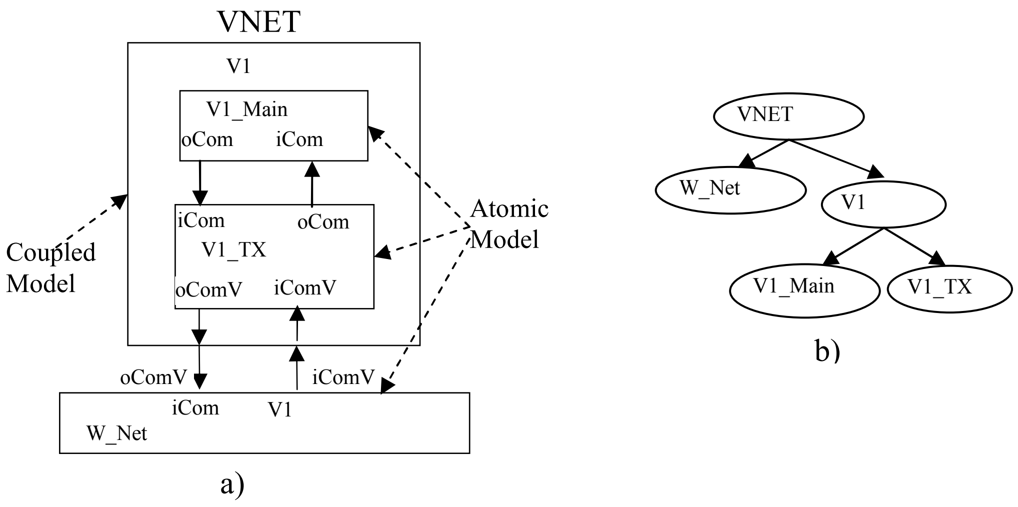


**Figure 7.1 a) Vehicular Wireless Communication Network: An example of coupling relation of DEVS formalism b) Hierarchical construction of VNET.**

**Dynamic Structure DEVS**

Many real systems have the characteristics to change their structure dynamically to adapt with the internal/external changes of the environment. Although DEVS is a popular method to simulate a variety of systems, it does not support changes in model structure during simulation run. The Dynamic Structure DEVS (DSDEVS) is the formalism to specify systems that can change their structure dynamically. The two well-accepted DSDEVS algorithms are Dynamic Structure Discrete Event (DSDE) system [58, 59] and dynDEVS [60]. The DSDE supports changes in structure by the introduction of a special model called *network executive* that stores all possible states of structural changes and their corresponding component sets in each structural state. Changes in executive state are automatically mapped into changes in structure. Alternatively, dynDEVS introduces two dynamic DEVS models: dynDEVS (atomic) and dynNDEVS (coupled) along with their corresponding model transition function, $\rho_\alpha$ (atomic) and $\rho_N$ (coupled). In dynDEVS, a model's state space, internal and external transition, output, time advance, and model transition functions are subject to change during simulation. Formally, a DSDEVS is represented by

$DSDEVS = < X, Y, S, s_0, \tau, \delta_x, \delta_y >$ , where $X$ and $Y$ are input and output event set, respectively, $S = S_{self} \times < D, \{M_i\}, C >$ is the set of partial states, $S_{self}$ is the set of self states, $< D, \{M_i\}, C >$ is the structure information, $C = \left( X \times \bigcup_{i \in D} X_i \right) \bigcup \left( \bigcup_{i \in D} Y_i \times \bigcup_{j \in D; j \neq i} X_j \right) \bigcup \left( \bigcup_{i \in D} Y_i \times Y \right)$ is the set of couplings, $s_0 \in S$ is the initial partial state, $\delta_x$ and $\delta_y$ are the external and internal transition functions, respectively.

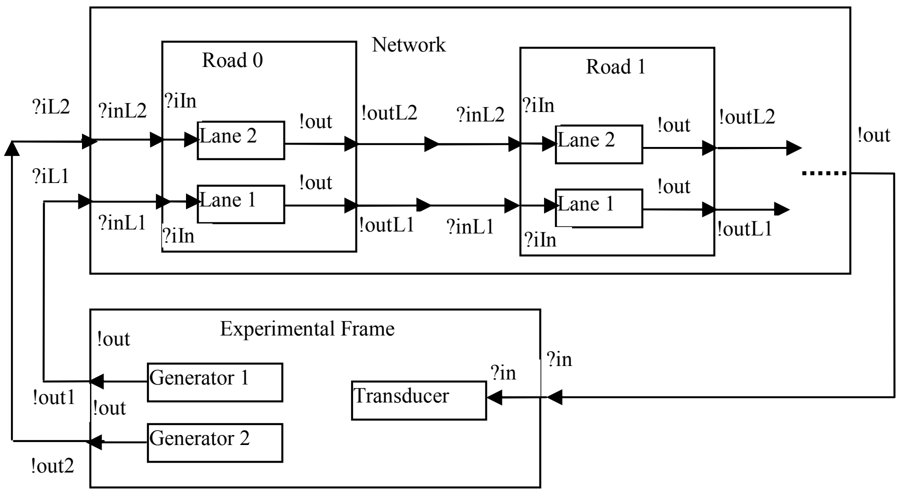
## DEVS Simulator

Ziegler proposed the abstract simulator concept for simulation of DEVS models [61, 62]. In abstract simulation algorithm, each model is associated with a virtual processor that interprets the dynamics specified by the formalism in a one-to-one manner.

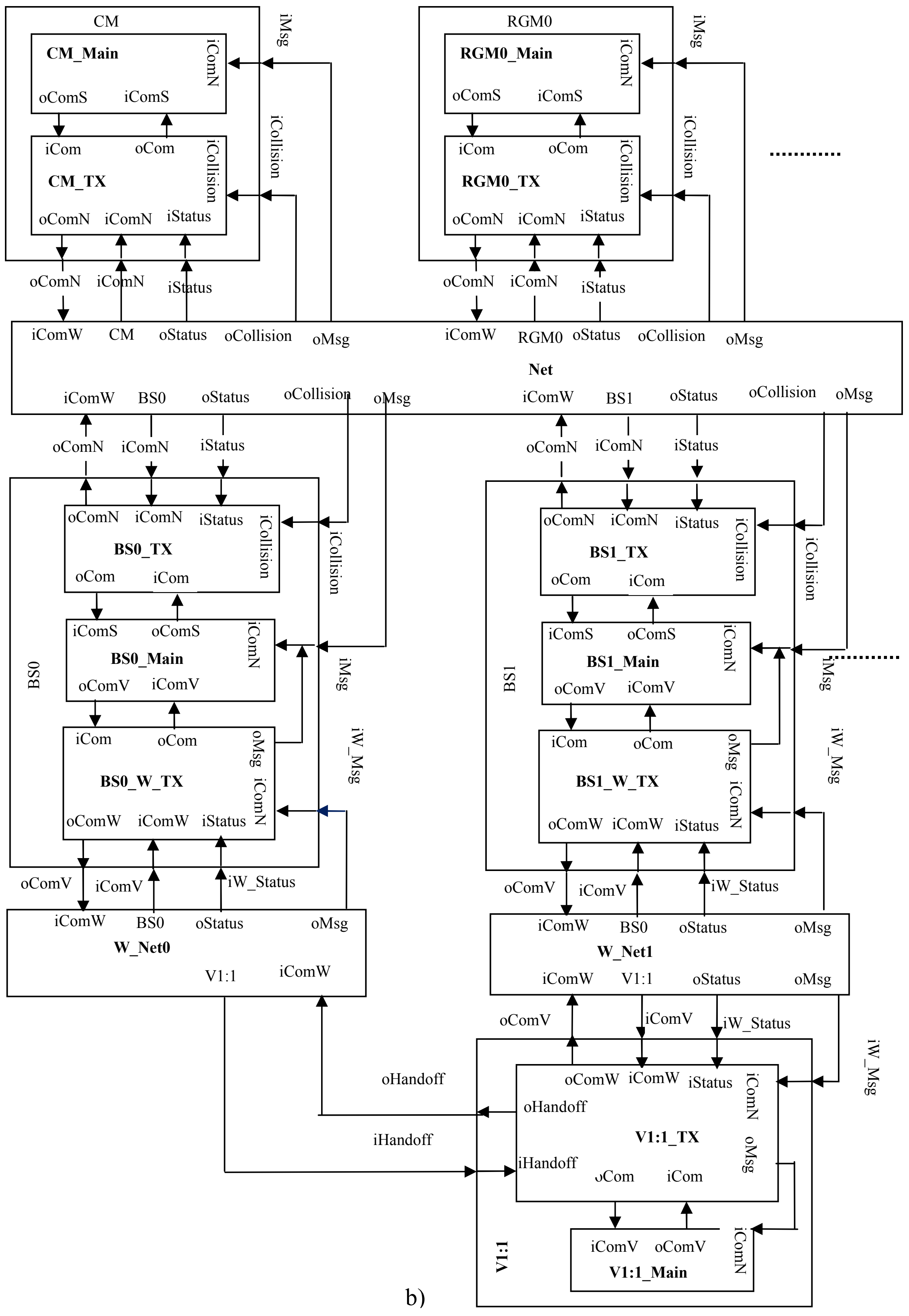
Simulation proceeds by means of message passing among the processors, not among DEVS models. The messages carry information about internal and external events as well as data needed for synchronization. There are two types of processors: a *simulator* for an atomic model and a *coordinator* for a coupled model. A special kind of coordinator called *root coordinator* which is not associated with any model, is responsible for advancing the simulation time. Each processor simulates a system by sending and/or receiving the four types of messages - *, x, y and done. The details of how the simulation runs can be found in [55, 61, 62].

**The DEVS based Simulation Model Development for VSDN**

The entities involved in the VSDN architecture are: CM, RGM, BS, Vehicle, Road, Net (Wired Communication Channel) and W_Net (Wireless Communication Channel). The DEVS architecture of this system requires 11 atomic models: CM_Main, RGM_Main, BS_Main, V_Main, TX, W_TX, Net, W_Net, Road, Generator and Transducer. The entities CM, RGM, BS and Vehicle are defined as coupled model consist of two or more atomic models mentioned above and Road is defined as DSDEVS. There is another coupled model called 'Experimental Frame (EF)' that consists of Generators and Transducer. The coupling relations of all the models are shown in Figure 7.2.


Network
Road 0
Road 1
?iL2
?inL2
?iIn
Lane 2
!out
!outL2
?inL2
?iIn
Lane 2
!out
!outL2
!out
?iL1
Lane 1
!out
Lane 1
!out
?inL1
?iIn
!outL1
?inL1
?iIn
!outL1
Experimental Frame
!out
Generator 1
?in
?in
Transducer
!out1
!out
Generator 2
!out2

a)


CM
CM_Main
iComN
oComS
iComS
iCom
oCom
CM_TX
iCollision
oComN
iComN
iStatus
iMsg
iCollision
oComN
iComN
iStatus
RGM0
RGM0_Main
RGM0_TX
iComW
CM
oStatus
oCollision
oMsg
RGM0
Net
BS0
BS1
BS0_TX
BS0_Main
BS0_W_TX
iComS
oComS
oComV
iComV
oMsg
iW_Msg
iW_Status
BS1_TX
BS1_Main
BS1_W_TX
W_Net0
W_Net1
V1:1
oHandoff
iHandoff
V1:1_TX
V1:1_Main
........

b)

**Figure 7.2 Complete VNET simulation Model. b) Experimental Frame coupled with Network a) Structure of Network Coupled Model that consists of other atomic and coupled models.**

# BIBLIOGRAPHY

1. S. Duri, "Framework for Security and Privacy in Automotive Telematics," in *2nd International Workshop on Mobile Commerce*. 2002.
2. J. S. Park and D. Dicoi, "WLAN Security: current and future," in *IEEE Journal on Internet Computing* 2003. **7**(5): p. 60-65.
3. W. Shunman, T. Ran., W. Yue and Z. Ji, "WLAN and it's Security Problems," in *Parallel and Distributed Computing, Applications and Technologies (PDCAT'2003)*. 2003.
4. L. H. Sahasrabuddhe and B. Mukherjee, "Multicast Routing Algorithms and Protocols: A tutorial," in *IEEE Network*, 2000: p. 90-102.
5. S. Rafaeli and D. Hutchison, "A survey of Key Management for Secure Group Communication," in *ACM Computing Surveys*, 2003. 35(3): p. 309-329.
6. D. Bruschi and E. Rosti, "Secure Multicast in Wireless Networks of Mobile Hosts: Protocols and Issues," in *Mobile Networks and Applications*, 2002. **7**: p. 503-511.
7. Y. Challal and H. Seba, "Group Key Management Protocols: A Novel Taxonomy," in *International Journal of Information Technology* (IJIT), 2005. **2**(1): p. 105-118.
8. IEEE Standard for Information and Technology - Telecommunications and Information Exchange between Systems, *Carrier Sense Multiple Access with*

*Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, 1985: New York.

9. R. M. Metcalfe and D. R. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks," in *Communications of the ACM*, 1976. 19(7): p. 395-403.

10. F. A. Tobagi and V. B. Hunt, "Performance Analysis of Carrier Sense Multiple Access with Collision Detection," in *Computer Networks*, 1980. 4: p. 245-259.

11. K. Sohraby, M. Molle and A. Venetsanopoulos, "Comments on "Throughput Analysis for Persistent CSMA Systems," in *IEEE Transactions on Communications*, 1987. COM-35(2): p. 240-243.

12. Y. Yasuda and I. Iida, "Performance Analysis of CSMA/CD Protocol with Backoff Algorithms," in *Electronics and Communications in Japan*, 1983. 66-B(10).

13. S. S. Lam, "A Carrier Sense Multiple Access Protocol for Local Networks," in *Computer Networks*, 1980. 4(21-32).

14. W. C. Chan. Random Access Networks, in *Performance Analysis of Telecommunications and Local Area Networks*, 2000, Springer. p. 399-445, ISBN 978-0-7923-7701-6.

15. IEEE Standard for Information Technology - Telecommunications and Information Exchange between Systems - Specific Requirements - Part 11: *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, *IEEE Std 802.11-1999 (ISO/IEC 8802-11:1999)*1999: New York.

16. G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," in *IEEE Journal on Selected Areas in Communications*, 2000. 18(3): p. 535-547.

17. P. Chatzimisios, A. C. Boucouvalas and V. Vitsas, "Performance Analysis of the IEEE 802.11 MAC Protocol for Wireless LANs," in *International Journal of Communication Systems*, 2005. 18: p. 545-569.

18. P. Chatzimisios, A. C. Boucouvalas and V. Vitsas, "Performance Analysis of IEEE 802.11 DCF in Presence of Transmission Errors," in *IEEE International Conference on Communications*2004. p. 3854-3858.

19. H. Zhai, Y. Kown and Y. Fang, "Performacne Analysis of IEEE 802.11 MAC Protocols in Wireless LANs," in *Wireless Communications and Mobile Computing*, 2004. 4: p. 917-931.

20. W Diffie and M. E. Hellman, "New Directions in Cryptography," in *IEEE Transactions on Information Theory*, 1976. IT-22: p. 644-654.

21. W. Diffie , P. C. Van Oorschot and M. J. Wiener, "Authentication and Authenticated Key Exchanges," in *Codes and Cryptography*. 2: p. 107-125.

22. *The Wireless Application Protocol Forum*.

23. WAP Forum, Wireless Application Protocol Architecture Specification. WAP-210-WAPArch-200100712-a, 12-July-2001.

24. R. J. Boncella, "Wireless Security: An Overview," in *Proceedings of the 8th Communication of the Association of Information Systems*. 2002.

25. S. Jormalainen and J. Laine, "Security in the WTLS," in Internet resource, [http://www.tml.tkk.fi/Opinnot/Tik-110.501/1999/papers/wtls/], 1999.

26. Markku-Juhani Saarinen, "Attacks Against the WAP WTLS Protocol," in *Proceedings of the IFIP TC6/TC11 Joint Working Conf. on Secure Information Networks: Communications and Multimedia Security*. 1999.

27. V. Gupta and S. Gupta, "Securing the Wireless Internet," in *IEEE Communication Magzine*, 2001. 39(12): p. 68-74.

28. P. G. Argyroudis, R. Verma, H. Tewari and D. O'Mahony, "Performance Analysis of Cryptographic Protocols on Handheld Devices," in *Proceedings on Third IEEE International Symposium on Network Computing and Applications, 2004, (NCA 2004)*. 2004.

29. G. Kambourakis, A. Rouskas and S. Gritzalis, "Using SSL/TLS in Authentication and Key Agreement Procedures of Future Mobile Networks," in *Proceedings of 4th International Workshop on Mobile and Wireless Communication Networks*. 2002.

30. G. Kambourakis, A. Rouskas and S. Gritzalis, "Experimental Analysis of an SSL-Based AKA Mechanism in 3G-and-Beyond Wireless Networks," in *International Journal of Wireless Personal Communications*, 2004. 29(3): p. 303 - 321.

31. W. A. Arbaugh, N. Shankar, Y. C. Justin Wan, "Your 80211 wireless network has no clothes," in *IEEE Journal on Wireless Communications*, 2002. 9(6): p. 44 - 51.

32. N. Borisov, I. Goldberg and D. Wagner, "Intercepting Mobile Communications: The Insecurity of 802.11," in *Proceedings of the 7th International Conference on Mobile Computing and Networking (Mobicom'01)*. 2001.

33. S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4," in *Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography (SAC '01)* 2001.

34. A. Stubblefield, J. Ioannidis and A. D. Rubin, "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP," in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2002*.

35. Wi-Fi Protected Access: Strong, Standard-based, Interoperable Security for Today's Wi-Fi Networks, Wi-Fi Alliance, April 29, 2003.

36. IEEE Standard for Local and Metropolitan Area Networks – *Port-Based Network Access Control*, *IEEE Standard 802.1X-2001*, 2001.

37. B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson and H. Levkowetz. *Extensible Authentication Protocol (EAP)*, 2004, RFC3748.

38. IEEE P802.11i/D10.0. Medium Access Control (MAC) Security Enhancements, Amendment 6 to IEEE Standard for Information Technology – Telecommunications and Information Exchange between Systems – Local and Metropolitan Area Networks – Specific Requirements – Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 2004.

39. C. He and J. C. Mitchell, "Security analysis and improvements for IEEE 802.11i," in *The 12th Annual Network and Distributed System Security Symposium (NDSS'05)*. 2005.

40. S. E. Deering, "Multicast Routing in Internetworks and Extended LANs," in *ACM SIGMOCOM*. 1988.

41. P. Judge and M. Ammar, "Security Issues and Sulutions in Multicast Content Distribution: A Survey," in *IEEE Network*2003. p. 30-36.

42. U. Varshney, "Multicast over Wireless Networks," in *Communication of the ACM*, 2002. 45(12): p. 31-37.

43. G. de Boer, P. Engel, W. Praefcke and Robert Bosch GmbH, "Generic Remote Software Update for Vehicle ECUs Using a Telematics Device as a Gateway," in *Advanced Microsystems for Automotive Applications 2005*, W.G. Jurgen Valldorf, Editor 2005, Springer. p. 371-380.

44. C. Y. Yeun and T. Farnham, "Secure Software Download for Programmable Mobile User Equipment," in *Third Generation Conference on 3G mobile Communication Technologies*. 2002.

45. S. Hirose and S. Yoshida, "An Authenticated Diffie-Hellman Key Agreement Protocol Secure Against Active Attacks," in *Public Key Cryptography*1998. p. 135 - 148.

46. S. Mittra, "Iolus: A framework for scalable secure multicasting," in *ACM SIGCOMM’97*, 1997.

47. R. Rivest, "The MD5 Message-Digest Algorithm," in *RFC 1321*1992.

48. V. Visoottiviseth, "Sender-Initiated Multicast for Small Group Communications," Nara Institute of Science and Technology, Japan, 2003.

49. D. R. Stinson. Cryptography: Theory and Practice. 3rd ed 2005: Boca Raton.

50. FIPS Publication 197, "Advanced Encryption Standard (AES)," U.S. DoC/NIST, 2001.

51. Y. Xiao, Hsiao-Hwa Chen, Bo Sun, R. Wang, and Sakshi Sethi, "MAC Security and Security Overhead Analysis in the IEEE 802.15.4Wireless Sensor Networks," in *EURASIP Journal onWireless Communications and Networking*, 2006: p. 1-12.

52. FIPS Publication 186, "Digital Signature Standard (DSS)," U.S. DoC/NIST, 2000.

53. H. Handschuh, P. Paillier, "Smart card Crypto-Coprocessors for Public-Key Cryptography," Lecture Notes in Computer Science, 2000. 1820: p. 386-394.

54. D. Gross, J. Shortly, J. Thompson, and C. M. Harris, Fundamentals of Queueing Theory (4th Edition), Willey Series in Probability and Statistics, 2008, New York: John Wiley & Sons, Inc.

55. B. P. Ziegler, H. Praehofer and T. G. Kim, Theory of Modeling and Simulation. 2nd ed. 2000: Academic Press, London.

56. M. H. Hwang, Modeling and simulation using DEVS\#, 2007.

57. S. Yousefi, E. Altman, R. El-Azouzi and M. Fathy, "Analytical Model for Connectivity in Vehicular Ad Hoc Networks," in *IEEE Transactions on Vehicular Technology*, 2008. **57**(6): p. 3341-3356.

58. F. J. Barros, "Modelling Formalisms for Dynamic Structure System," in. *ACM Transactions on Modeling and Computer Simulation*, 1997. **7**(4): p. 501 - 515.

59. F. J. Barros, "Abstract Simulators for the DSDE Formalism," in *Proceedings of 1998 Winter Simulation Conference*. 1998. Washington, DC , USA

60. A. M. Uhrmacher, "Dynamic Structures in Modeling and Simulation: a Reflective Approach," in *ACM Transactions on Modeling and Computer Simulation* (TOMACS), 2001. 11(2): p. 206 - 232.

61. B. P. Zeigler, Multifacetted Modelling and Discrete Event Simulation. Academic Press, 1984.

62. B. P. Zeigler, "Discrete Event Formalism for Model based Distributed Simulation," in *Proceedings of SCS Conference on Distributed Simulation, 1985*. San Diego, CA.

# ABSTRACT

# REMOTE SOFTWARE UPLOAD TECHNIQUES IN FUTURE VEHICLES AND THEIR PERFORMANCE ANALYSIS

by

**IRINA HOSSAIN**

May 2012

**Advisor:** Dr. Syed Masud Mahmud

**Major:** Computer Engineering

**Degree:** Doctor of Philosophy

Updating software in vehicle Electronic Control Units (ECUs) will become a mandatory requirement for a variety of reasons, for examples, to update/fix functionality of an existing system, add new functionality, remove software bugs and to cope up with ITS infrastructure. Software modules of advanced vehicles can be updated using Remote Software Upload (RSU) technique. The RSU employs infrastructure-based wireless communication technique where the software supplier sends the software to the targeted vehicle via a roadside Base Station (BS). However, security is critically important in RSU to avoid any disasters due to malfunctions of the vehicle or to protect the proprietary algorithms from hackers, competitors or people with malicious intent. In this

thesis, a mechanism of secure software upload in advanced vehicles is presented which employs mutual authentication of the software provider and the vehicle using a pre-shared authentication key before sending the software. The software packets are sent encrypted with a secret key along with the Message Digest (MD). In order to increase the security level, it is proposed the vehicle to receive more than one copy of the software along with the MD in each copy. The vehicle will install the new software only when it receives more than one identical copies of the software. In order to validate the proposition, analytical expressions of average number of packet transmissions for successful software update is determined.  Different cases are investigated depending on the vehicle's buffer size and verification methods. The analytical and simulation results show that it is sufficient to send two copies of the software to the vehicle to thwart any security attack while uploading the software.

The above mentioned unicast method for RSU is suitable when software needs to be uploaded to a single vehicle.  Since multicasting is the most efficient method of group communication, updating software in an ECU of a large number of vehicles could benefit from it.  However, like the unicast RSU, the security requirements of multicast communication, i.e., authenticity, confidentiality and integrity of the software transmitted and access control of the group members is challenging. In this thesis, an infrastructure-based mobile multicasting for RSU in vehicle ECUs is proposed where an ECU receives the software from a remote software distribution center using the road side BSs as gateways. The Vehicular Software Distribution Network (VSDN) is divided into small regions administered by a Regional Group Manager (RGM). Two multicast Group Key

Management (GKM) techniques are proposed based on the degree of trust on the BSs named Fully-trusted (FT) and Semi-trusted (ST) systems. Analytical models are developed to find the multicast session establishment latency and handover latency for these two protocols. The average latency to perform mutual authentication of the software vendor and a vehicle, and to send the multicast session key by the software provider during multicast session initialization, and the handoff latency during multicast session is calculated. Analytical and simulation results show that the link establishment latency per vehicle of our proposed schemes is in the range of few seconds and the ST system requires few ms higher time than the FT system. The handoff latency is also in the range of few seconds and in some cases ST system requires less handoff time than the FT system. Thus, it is possible to build an efficient GKM protocol without putting too much trust on the BSs.

# AUTOBIOGRAPHICAL STATEMENT

Mrs. Irina Hossain received her M.Sc. degree in Electrical and Computer Engineering from University of Manitoba, Canada, in 2004 and the B.Sc. degree in Electrical and Electronic Engineering from Bangladesh University of Engineering and Technology (BUET) in 2000. She is currently working at Mayo Clinic, Rochester, MN as an IT professional. She was a Teaching Associate at Wayne State University from 2004 to 2006. Prior to that, she was a lecturer at BUET. She received Graduate Professional Scholarship (GPS) award at Wayne State University in 2006 - 2007. She received University of Manitoba Graduate Fellowship (UMGF) in the year of 2002 – 2003. Her research interests include wireless multicasting, wireless communication in vehicular network and security in vehicular network. Her recent publications include:

1. I. Hossain, S. M. Mahmud and M. Ho Hwang, "Performance Evaluation of Mobile Multicast Session Initialization Techniques for Remote Software Upload in Vehicle ECUs," *Proceedings of the 2010 IEEE 72nd Vehicular Technology Conference: VTC2010-Fall*, September 6–9, 2010, Ottawa, Canada.
2. I. Hossain and S. M. Mahmud, "Analysis of Group Key Management Protocols for Secure Multicasting in Vehicular Software Distribution Network," *Proceedings of the Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2007)*, Crowne Plaza Hotel, White Plains, New York, USA, October 8-10, 2007.
3. I. Hossain and S. M. Mahmud, "Analysis of a Secure Software Upload Technique in Advanced Vehicles using Wireless Links," *Proceedings of the 2007 IEEE Intelligent Transportation Systems Conference*, Seattle, WA, USA, Sept. 30 - Oct. 3, 2007, pp. 1010-1015.
4. I. Hossain and S. M. Mahmud, "Secure Multicast protocol for remote software upload in intelligent vehicles," *Proc. 5th Annual Intelligent Vehicle System Symposium National Defense Industries Association (NDIA)*, National Automotive Center and Vectronics Technology, pp. 145-155, June 13 –16, 2005, Traverse City, Michigan.
5. S. M. Mahmud, S. Shanker and I. Hossain, "Secure Software Upload in an Intelligent Vehicle via Wireless Communication Links," *Proc. 2005 IEEE Intelligent Vehicles Symposium*, June 6-8, 2005, Las Vegas, Nevada, USA.
6. W. Khalid, I. Hossain, S. M. Mahmud and Y. Xu, "Intelligent Vehicle Based Architecture for Real-Time Monitoring of Soldiers' Health using MEMS Flexible Smart Skin Sensors," *Proc. 5th Annual Intelligent Vehicle Systems Symposium of National Defense Industries Association (NDIA)*, National Automotive Center and Vectronics Technology, pp. 59-64, June 13 –16, 2005, Traverse City, Michigan.