A SECURE PREPAID MICROPAYMENT SCHEME FOR WIRELESS MESH NETWORKS

by
CAN YÜCEL

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
August 2010

A SECURE PREPAID MICROPAYMENT SCHEME FOR WIRELESS MESH NETWORKS

APPROVED BY

Assoc. Prof. Dr. Albert Levi ....

(Thesis Supervisor)

Asst. Prof. Dr. Cemal Yılmaz ....

Assoc. Prof. Dr. Erkay Savaş ....

Assoc. Prof. Dr. Özgür Erçetin ....

Assoc. Prof. Dr. Yücel Saygın ....

DATE OF APPROVAL ....

A SECURE PREPAID MICROPAYMENT SCHEME FOR WIRELESS MESH NETWORKS

Can Yücel
Computer Science and Engineering, MS Thesis, 2010
Thesis Supervisor: Assoc. Prof. Albert Levi

Keywords: Prepaid Payment Systems, Security, Micropayments, Wireless Mesh Networks

Abstract

Wireless Mesh Network (WMN) is an evolving multi-hop, ubiquitous and high speed networking technology. In this thesis, we proposed a secure micropayment scheme for network access in WMNs. The main motivation is that the operators are not considered as fully trusted entities in our scheme; the clients control their balance with their operators. In this way, none of the system entities can behave dishonestly about the amount of services provided and obtained.

Our proposed payment scheme is a prepaid one. The users obtain connection cards for getting service. Connection card issuer, which is a trusted third party, generates the connection cards. Each connection card includes tokens. These tokens are generated as a hash chain which is obtained by hashing an initial value (IV) several times. Hash functions are one way and irreversible cryptographic functions. The tokens are consumed backwards. Therefore, it is not feasible to generate unused tokens from an already used token. This property is the main enabler for the security of our scheme.

We have conducted simulations for performance evaluation of the proposed scheme. Our results show that in a network with 300 clients, the average authentication completion time becomes less than 1 second even if all the clients send their connection request at the same time.

KABLOSUZ ÖRGÜ AĞLARI İÇİN ÖN ÖDEMELİ GÜVENLİ MİKROÖDEME ŞEMASI

Can Yücel
Bilgisayar Bilimi ve Mühendisliği, Yüksek Lisans Tezi, 2010
Tez Danışmanı: Doç. Dr. Albert Levi

Anahtar Kelimeler: Ön Ödemeli Sistemler, Güvenlik, Mikro Ödeme, Kablosuz Örgü Ağlar

Özet

Kablosuz Örgü Ağlar çok sekmeli, erişimin her yerden sağlanabildiği, yüksek hızlı ve gelişmekte olan bir ağ teknolojisidir. Bu tezde kablosuz örgü ağlarda ağ erişimi için güvenli bir mikro ödeme sistemi teklif edilmiştir. Şemamızdaki ana motivasyon, operatörlerin tamamen güvenilir birimler olmadığı yönündedir; buna göre müşteriler operatörlerle olan hesaplarını kendileri kontrol altına almaktadır. Bu sayede, sistem elemanlarının hiç birisi sağlanan veya alınan hizmet karşısında dürüst olmayan bir davranış sergileyememektedir.

Önerdiğimiz ödeme sistemi ön ödemeli bir şemadır. Kullanıcılar bağlantı kartları sayesinde servis alırlar. Bağlantı kartları düzenleyicisi, güvenilir bir üçüncü parti kuruluştur ve bağlantı kartlarının yaratılmasından sorumludur. Her bir bağlantı kartı jetonlardan oluşur. Bu jetonlar özet zinciri olarak yaratılır. Özet zinciri başlangıç değerinin bir kaç defa özetlenmesi ile oluşur. Özet fonksiyonları tek yönlü ve geri alınamayan kriptografik fonksiyonlardır. Jetonlar ise geriye doğru tüketilir. Bu sayede kullanılmış jetonlardan kullanılmamış jetonların üretilmesi mümkün değildir. Bu özellik şemamızdaki güvenliği sağlayan ana özelliktir.

Önerdiğimiz şemanın performans değerlendirmesi için simülasyonlar gerçekleştirdik. 300 müşteri ile elde ettiğimiz sonuçlar tüm müşteriler bağlantı isteğini aynı anda gönderdiğinde ortalama kimlik doğrulama süresinin 1 saniyenin altında olduğunu göstermiştir.

To my family

Acknowledgements

I would like to thank to my thesis advisor, Albert Levi, for all of his assistance on my thesis with flexible working hours (which includes meetings after all day work), for his guidance with every issue in whole my Mastering Degree life, and most importantly for his ever-lasting patience.

I would like to thank Erkay Savaş, for his great assistance on developing myself in academical life with all the feedbacks he has given in his courses.

As his teaching assistant, I thank Cemal Yılmaz, for caring all our opinions in his course, and sharing his knowledge on Software Engineering.

I also thank Yücel Saygın, and Özgür Erçetin for devoting their time amongst their high volume schedule and joining my jury.

Even I know they will never have a look at my thesis, and will not read this, I specially thank my family with all my heart for all their efforts while I am working on my thesis, and for their full trust and support on me in whole my life.

And thanks to all my friends who have been bored of hearing, “I cannot go out tonight, because I have to work on my thesis”, but never complain about this –at least not that much.

I also thank Scientific and Technological Research Council of Turkey (TÜBİTAK) for funding me by BİDEB scholarship.

TABLE OF CONTENTS

1. INTRODUCTION 1

1.1. Contribution of the Thesis 1

1.2. Organization of the Thesis 3

2. BACKGROUND ON WIRELESS MESH NETWORKS 4

2.1. Components of WMN 5

2.2. Wireless Mesh Network Types 7

3. BACKGROUND ON CRYPTOGRAPHIC PRIMITIVES 9

3.1. Hash Functions 9

3.2. Hash Chains 10

3.3. HMAC 11

3.4. Symmetric Cryptography 12

3.5. Public Key Cryptography 12

4. PROPOSED SECURE PREPAID PAYMENT SCHEME 15

4.1. Requirements of a Payment System 15

4.2. General Overview of the Proposed Scheme 16

4.3. Network Entities and Assumptions 18

4.4. Connection Card Issuer 20

4.4.1. Connection Cards 20

4.4.2. Creation of Tokens 21

4.4.3. Distribution of the Connection Card Information to the Operators 22

4.5. Authentication 23

4.5.1. Initial Authorization..24

4.5.2. Reuse of a Connection Card...27

4.5.3. Access Point Authentication 29

4.6. Packet Transfer...30

4.7. Roaming..31

4.8. Update Packets and Disconnection..34

4.8.1. Update Packets 34

4.8.2. Disconnection..36

4.9. Seamless Mobility in Home Operator 39

4.10. Settlement among the Operators 41

4.11. Discussion 43

5. SIMULATION DETAILS AND PERFORMANCE EVALUATION.........45

5.1. Simulation Details 46

5.1.1. Environmental Details 46

5.1.2. Public Key Operations and Their Timings...48

5.1.3. Storage Requirements...49

5.1.4. Performance Metrics...50

5.2. Simulation Scenarios and Results..51

5.2.1. Burst Scenario 51

5.2.2. Real-Life Scenario...54

5.2.3. Rush Scenario 57

5.2.4. Mobile Scenario 58

6. CONCLUSIONS 61

7. REFERENCES 63

LIST OF FIGURES

Figure 2.1. Conventional Access Point Communication 5

Figure 2.2. Mesh Backbone [1]... 6

Figure 2.3. Coverage Area of Wi-Fi and WMN [8].. 7

Figure 3.1. Encryption with Public Key Cryptography [7].................................13

Figure 3.2. Authentication [7]..14

Figure 4.1. Network Entities...18

Figure 4.2. Initial Authorization 24

Figure 4.3. Reuse of a Connection Card...27

Figure 4.4. Access Point Authentication Using a Challenge-Response Protocol..29

Figure 4.5. Packet Transfer..30

Figure 4.7. Update Packets 35

Figure 4.8. Disconnection (Home Operator) 36

Figure 4.10. Mobile Client 40

Figure 5.1. Clustered Structure 47

Figure 5.2. Gateway backbone...47

Figure 5.3. Simulation Environment (*AP*s and Servers) 48

Figure 5.4. End-to-End Connection Establish Time in Burst Scenario (Home Network)..52

Figure 5.5. Server Service Delay 53

Figure 5.6. End-to-End Connection Establish Time in Rush Scenario (Foreign Network)..53

Figure 5.7. Average Connected Client Count per Access Point...........................54

Figure 5.8. End-to-End Delay in Real-Life Scenario (Home Network) 56

Figure 5.9. Connected Node Count Details (Home Network).............................56

Figure 5.10. End-to-End Delay in Real-Life Scenario (Foreign Network)...........57

Figure 5.11. End-to-End Delay in Rush Scenario...58

Figure 5.12. End-to-End Delay in Mobile Scenario 59

Figure 5.13. Connected Node Count Details in Mobile Scenario.........................60

LIST OF TABLES

Table 4.1. Symbols used in payment scheme 19

Table 4.2. Connection Card Details 21

Table 5.1. AP Specifications..46

Table 5.2. Platform Specifications 49

Table 5.3. 1024 bit RSA timings..49

Table 5.4. Storage requirements for each client 50

1. INTRODUCTION

Wireless Mesh Network (WMN) is a multi-hop wireless networking technology to provide broadband ubiquitous access in metropolitan area. WMN provides flexible service like ad hoc networks, and supports this flexibility with the centralized structure of Wi-Fi. With the assistance of wireless mesh routers, WMNs are easily established. Moreover, WMN provide high connection speeds with less overhead.

With the potential number of active users, it is inevitable that there will be more than one operator, who will charge the general use of the clients and provide connection in the metropolitan area. Current payment schemes mostly consider that the operators are fully trusted, and clients are potential cheaters. However in real life, the operators may intentionally or unintentionally overcharge their users.

1.1. Contribution of the Thesis

In this thesis, we propose a prepaid payment scheme for WMNs. Our scheme aims to give the control to the clients. Moreover, we designed protocols for avoiding cheating among clients and the operators.

In our proposed system, for providing an efficient prepaid payment scheme in ubiquitous networks, we are inspired in making micropayments which includes tokens in a connection card. The tokens are generated using cryptographic hash chains [3]. In this scheme an initial value (IV) is hashed n times and each of the computed elements serves as tokens. While getting service, these tokens are submitted to the operator beginning from the first, $H_0 = h^n(IV)$, to the last token, $H_n = h(IV)$. Hash functions are one way and irreversible functions. Hence, when a token is submitted by the client, it is not possible to derive unused tokens from this submitted one without knowing *IV*.

In our payment scheme, the only fully trusted entity is *connection card issuer* (*CCI*). It is responsible for generating connection cards for the clients. After generation of connection cards, *CCI* only distributes first token of each connection card to the operators. Moreover, operators get information of the used tokens from the clients while providing service. Hence, an operator cannot charge a client for unprovided service.

For mutual authentication and authorization among clients and operators, we have designed several protocols. With the use of these protocols and the token mechanism described above, the clients cannot repudiate using a token for services obtained and spend a token twice.

In our scheme it is possible to roam between the operators using the same connection card. All the used tokens are stored in the operators' databases. Hence for the settlement phase an operator submits the used tokens of the foreign operator for getting paid.

We have simulated our network using Omnet++ [11]. The performance metrics we use are end-to-end authentication latency and server service delay. For performance evaluation we consider two scenarios: one is a stress test, the other is typical real life scenario. In stress test 300 clients simultaneously send connection requests to server. Requests are handled in 669 ms for the home operator and 678 ms for foreign operator. This shows even in a rush scenario, our protocol handles connection requests in less than 1 second. For real life scenario, after sending connection requests, clients start TCP sessions. In this scenario for 40 clients, average end-to-end authentication latency is 532 ms for home operator. Standard deviation of this latency is 409 ms. The reason of this high standard deviation value is, with network congestion, some of the connection request packets are dropped. Therefore, connection request packets are resend. This shows us even congestion occurs in the network, our protocol has a considerable authentication mean.

1.2. Organization of the Thesis

The rest of this thesis is organized as follows. In Section 2, we give background on Wireless Mesh Networks. In Section 3, we explained background information on cryptographic schemes that our protocol is built upon. In Section 4, we give details of our proposed scheme. Section 5 gives the results of our simulation and performance analysis. Finally, Section 6 presents the conclusions.

2. BACKGROUND ON WIRELESS MESH NETWORKS

Since the last decade, wireless technologies has been evolved. In the beginning of new millennium with expanding technology of GSM, first Wireless Application Protocol (WAP) is announced, and then respectively GPRS and 3G have followed this technology. Apart from GSM, while some existing networks like Wi-Fi and sensor networks are advancing, some new wireless technologies like WiMAX [20], and Wireless Personal Area Networks (WPANs) are arising. Nowadays, current wireless network technologies that are commonly used are Wi-Fi and ad hoc networks. These two network types have some significant drawbacks, if we consider providing connection service to a metropolitan area.

Ad hoc networks are decentralized network structures that every node participates in routing protocols by forwarding received packets. Since it does not need a pre-established wired backbone, it is easy to form a network with active nodes. Moreover since it does not comprise any stationary communication unit, an ad hoc network just covers a small area. Due to lack of a static structure, if a node drops out, some links between the nodes could easily be broken. This decentralized architecture is very suitable for some military applications, but for a metropolitan area, it is inadequate.

On the other hand, Wi-Fi hotspots provide centralized structure, and it is adequate for connecting nodes in a city. The problem arises when we attempt to cover all the areas that the nodes are stationary. Since the access points (APs) cannot communicate with each other, we have to connect all APs with wires. Hence, this increases overhead of the networking establishment. Since the nodes cannot communicate with each other, this causes APs to use conventional star type communication as shown in Figure 2.1. In this topology all the nearby stations are connected to a single hub. Hence, network traffic sometimes encounters with bottlenecks.

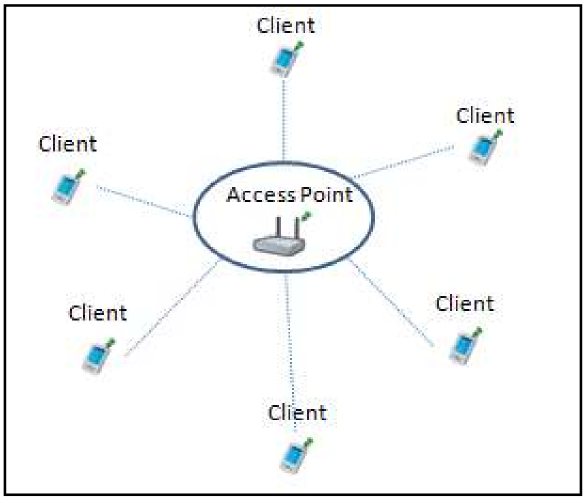

Figure 2.1. Conventional Access Point Communication

At this point, researchers focused on WMNs [1]. Basically WMNs have central infrastructure of Wi-Fi, and flexibility of ad hoc networks. As in the traditional Wi-Fi networks with the centralized structure, WMNs are basically based on gateways, access points (APs) and clients.

The differences between WMN and conventional ad hoc and Wi-Fi networks are the following:

- WMNs have been more flexible with the implementation of the modified versions of ad hoc routing protocols supporting communication between the entities in a wireless environment.
- With the participation of the nodes and APs in routing protocols, network is more transparent to these entities.

2.1. Components of WMN

The components of WMNs are mesh routers and mesh clients [1][17]. Mesh routers are a combination of gateways and access points with additional features. These routers have minimal mobility. Some mesh routers with gateway property can gain access to the backhaul network/Internet as shown in Figure 2.2. Apart from that, because mesh structure is not wired, it is easier to extend the network area with the same transmission power by establishing new mesh routers in an area. This is enabled because of the multi-hop structure.

Mesh routers also can be used as bridge for establishing connectivity of different networks. As shown in Figure 2.2, Wi-Fi, cellular, WiMAX and sensor networks can be connected with the mesh infrastructure via mesh routers. On the other hand, wireless or wired stations can also connect to mesh routers directly.

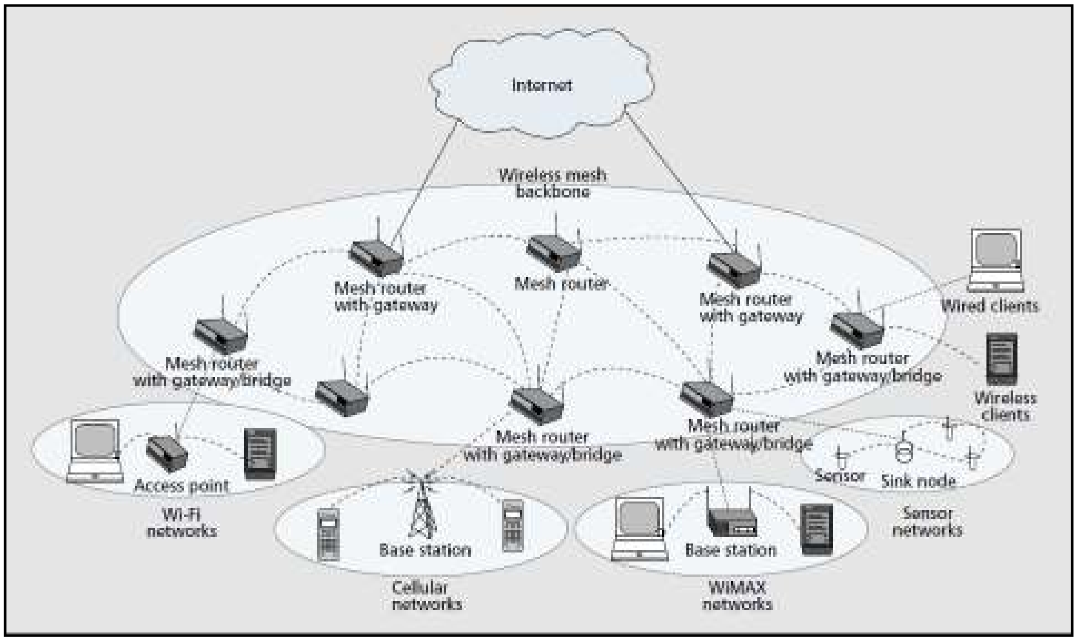

Figure 2.2. Mesh Backbone [1]

Mesh clients are regular stations, except that they can join routing protocols, which makes mesh backbone transparent for the nodes. Moreover, they have simple hardware compared to the mesh routers.

2.2. Wireless Mesh Network Types

There are three types of WMNs [1][8]: client-mesh network, infrastructure type mesh network, and hybrid mesh networks.

Infrastructure Mesh Network: In this mesh network type, mesh routers with gateway capabilities provide connection between the network and the backbone. All the mesh routers are connected to each other in a wireless medium. Mesh routers with bridging capabilities are responsible for the integration of other kind of networks as cellular or ad hoc networks with the mesh infrastructure.

This infrastructure type makes WMN suitable for many networking scenarios. One of the closest network type to end users is home networking. Current challenge of Wi-Fi networks in home networking is that APs are not being able to cover the dead zones in a house. At some points some devices will not have enough transmission power for connection. In such cases, the coverage should be extended by establishing new APs, which needs cable installation. In WMNs, by using mesh routers, all the users in a house can easily establish connection with each other. One of the routers could provide connection of all these devices to backhaul network.

The other example is community and neighborhood networking. All the networks that are established in the home networking could be connected with each other with the assistance of mesh routers. These mesh routers are placed on the roofs of the houses. In Figure 2.3, differences of the coverage zones between Wi-Fi and WMN in a neighborhood are depicted.

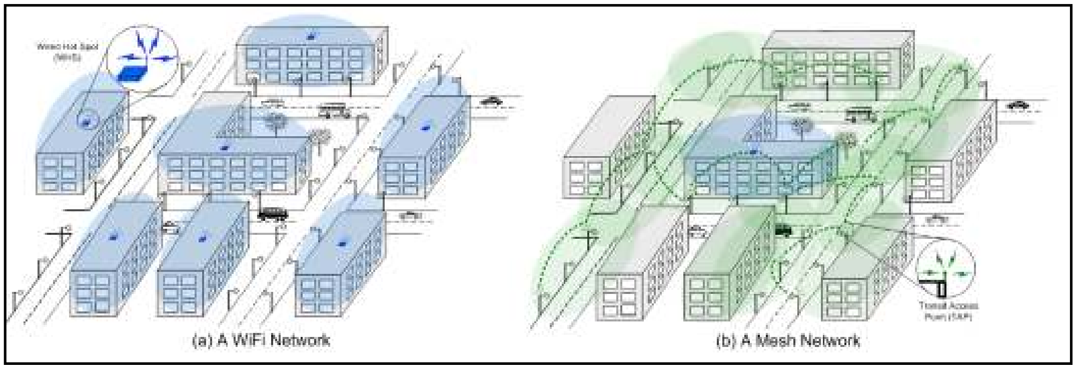

Figure 2.3. Coverage Area of Wi-Fi and WMN [8]

Client Mesh Network: Client mesh networks are established similar to conventional ad hoc networks with peer-to-peer connections of the nodes. In this type of connections, there is no need for a mesh router, but all the devices must have the same type of radio.

Hybrid Type Networks: This is the combination of infrastructure and client mesh networks. This combination is important especially in the home networking, because dead zones can easily be covered by using ordinary wireless devices. Clients can communicate with each other, while mesh routers are providing connectivity of the clients with the Internet backbone. Hence, it decreases the traffic burden of APs in conventional Wi-Fi.

3. BACKGROUND ON CRYPTOGRAPHIC PRIMITIVES

In this section, we give cryptographic protocols that we used in our payment scheme. In Section 3.1, we mention about hash functions. In Section 3.2 we give details of hash chains, which is the protocol that our scheme is based on. In Section 3.3 we give message authentication code background, and mention about HMAC. In Section 3.4 we give brief information about symmetric encryption. In Section 3.5 we mention about public key cryptography.

3.1. Hash Functions

Hash function [6][7] h is a one way and irreversible function that takes large variable size input and returns fixed and usually smaller size output. Because of their main features, they have taken very critical roles in security area. Main properties can be denoted as:

1. Input texts can be variable size, but h has a fixed size output.
2. h is an easily computable function.
3. h is a one-way and irreversible function, which makes computation of x infeasible from a given $h(x) = y$. This property is called weak collision resistance.
4. It is computationally infeasible to find colliding $h(x) = h(y)$ such that $x \neq y$. This property is called strong collision resistance.

3.2. Hash Chains

Hash chains [3] are established as applying a hash function on another hash value, which is shown as $h(h\left(h(\ldots h(x))\right))$, where *x* is a variable size input.

Hash chain with 3 elements is denoted as:

$$h^3(x) = h(h(h(x))) \tag{3.1}$$

More generally, hash chain with *n* elements is denoted as:

$$\underbrace{h\left(h(h(\ldots h(x) \ldots))\right)}_{n} = h^n(x) \tag{3.2}$$

In a hash chain based security protocol, derived hash values are used backwards as $h^n(x)$, $h^{n-1}(x)$, $h^{n-2}(x)$, …, $h(x)$. For instance an input value *x* is hashed 100 times, $(h^{100}(x))$, and this hash value is sent to a receiver. Then, *x* is hashed 99 times, $(h^{99}(x))$, and hash output is sent. This procedure proceeds until *x* is hashed only once, $(h(x))$. As we mention in previous section, hash functions are one way and irreversible functions. Hence, whenever an adversary eavesdrop a single hash value from the hash chain, it is not possible to derive following hash values; it can only derive the previous ones, which are already used and staled.

As another advantage of hash chains is that when two hash values are sent consecutively, second hash value can be validated by hashing it and comparing with the previous one. For instance, first $h^{35}(x)$, and then $h^{34}(x)$ is sent by a sender. When receiver gets these two values, he hashes the second one as $h(h^{34}(x))$. After that he compares whether it is identical with the previously sent hash value, $h^{35}(x)$. If they are equal, then $h^{34}(x)$ is validated. This validation feature of hash chains makes it unnecessary to store all the hash values in the receiver's storage area. For the receiver, storing only the last received hash value is adequate.

Hash functions are easily computational functions. This makes hash chains suitable for the security protocols. Hash chains are mostly used in creation of one-time keys from a single password, because of its irreversible output property. In each connection session, instead of using a new key, session keys are derived using hash chains over a previously decided password with the server. Each time a new session is started, a newly derived hash value is used as session key. Hence, even a hash value is received by an adversary; it is not possible for him to use this value in another session.

When all the hash values in a hash chain are used, a new initial value must be chosen.

3.3. HMAC

Message Authentication Codes (MACs) [6][7] are used for authentication of the messages, and generally used between two entities. This authentication is provided by the MAC functions. A message is given as the input of a MAC function, and an output *message authentication code* is produced from this message. MACs are formed by using a secret key.

Sender sends the message m, with its authentication code, which is generated by MAC function with the secret key *K*. Hence, the authentication code is: $MAC_K(m)$. For validation of a received message m, receiver gives the message as the input of the MAC function, and uses the same secret key, *K*. If the output of this functions is same with $MAC_K(m)$, then the message is authenticated.

HMACs [18] use hash functions for generating MACs. HMAC is generated as:

$$HMAC_K(m) = h(\ (K^+ \oplus opad) \parallel h((K^+ \oplus ipad) \parallel m)) \qquad (3.3)$$

HMAC formula is described as follows:

1. Secret key *K*, is appended with zeroes to the left for the creation of *b*-bit K^+. *b* denotes number of bits in a block.
2. *ipad* is a *b* bit value, and formed by repeating $(36)_{16}$ *b*/8 times. K^+ is XORed with *ipad*. The result is S_i.

3. *m* is concatenated with S_i. Afterwards, it is hashed. H_i is formed.
4. *opad* is a *b* bit value, and formed by repeating $(5C)_{16}$ *b*/8 times. K^+ is XORed with *opad*. The result is S_0.
5. S_0 is concatenated with $H_{i.}$ Afterwards, it is hashed.

As we have mentioned in Section 3.1, hash is a one-way function. Hence, when it is used by a secret key, it is not possible for an attacker to alter a message and generate a new fake HMAC without knowing the key.

3.4. Symmetric Cryptography

Symmetric cryptography is the cryptographic method that encryption and decryption is handled by the same secret key. This key is shared between two parties.

There exist two types of symmetric cryptography: stream cipher, and block cipher. Stream ciphers [21] encrypt a text one byte at a time. RC4 is the most popular stream cipher, which is used in *SSL* (Secure Socket Layer) and *WEP* (Wired Equivalent Privacy).

Block ciphers handle an input text as fixed size blocks, and produces equal length output ciphertext blocks. Most popular block cipher algorithm is Data Encryption Standard (*DES*) [23]. Other common block cipher algorithms are Advanced Enryption Standard *AES* [24], *Blowfish* [22], and *Triple DES* [23].

3.5. Public Key Cryptography

Main inspiration of public key cryptography (*PKC*) [5] lies behind using a pair of keys. One is publicly known public key. The other is secretly kept by the owner, which is known as private key. In *PKC*, it is computationally infeasible to retrieve private key from public key. Some *PKC* protocols are Diffie-Hellman key exchange protocol [25], Digital Signature Algorithm (*DSA*) [25], Elliptic Curve Cryptography (*ECC*) [26], and *RSA* [4].

PKC is used for four main operations. These are encryption, decryption, signature and verification. Encryption is for providing data integrity and confidentiality of a message. Hence, a message is encrypted with the public key of the receiver. Decryption of this message is handled by the private key of the receiver. For protecting the authenticity of a message, message is encrypted with the private key of the sender, and a signature is produced. Because the private key is secretly kept by the owner, this is known as signature operation. Verification is handled by the public key of the sender.

Encryption and decryption operations are shown in Figure 3.1. In this figure each user has a public-private key pair, which the public keys are shared with the other users, and private keys are kept secret by the owners.

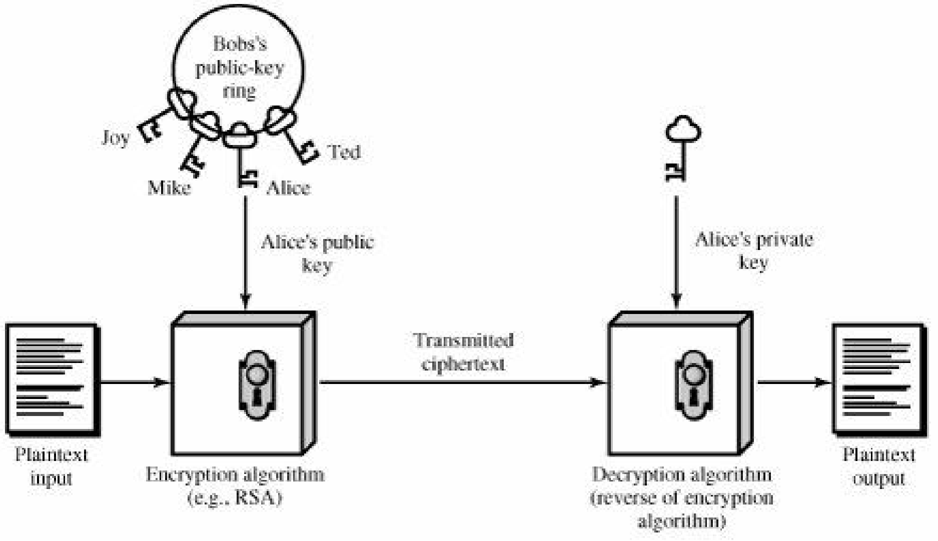

Figure 3.1. Encryption with Public Key Cryptography [7]

In Figure 3.1 Bob owns public keys' of Joy, Mike, Alice and Ted. He has encrypted the plaintext with Alice's public key, and a ciphertext is generated. This ciphertext can only be decrypted by Alice's private key.

Digital signature methods [25] are based on public key cryptography. In Figure 3.2 before signing operation, Bob generates its public-private key pair, and distributes its public key to the other system users. He seals the input text with his private key. This text can be authenticated by all the users who wields the public key of Bob (in Figure 3.2, Alice possesses it). Hence, as long as Bob keeps secret its private key, it is computationally infeasible to get this key for an adversary, and substitute Bob.

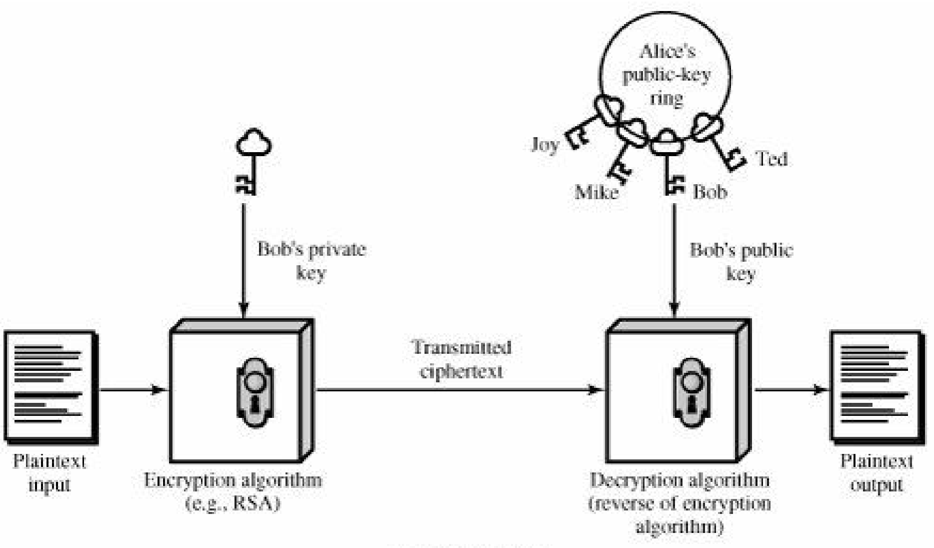

Figure 3.2. Authentication [7]

4. PROPOSED SECURE PREPAID PAYMENT SCHEME

In this section, we give design details of our proposed scheme. Before giving the details of the system, we mention about the requirements of a secure payment system for Metropolitan Area Networks. At the end of this section, we give which requirements are met.

4.1. Requirements of a Payment System

For Metropolitan Area Networks, requirements of a secure payment scheme are as follows:

- Wide Coverage: getting service without any restrictions in a larger zone is important for the users. With wireless routers, WMN has a suitable and flexible infrastructure for supporting wide area coverage.
- Roaming: users must be able to connect in all operators' zone by using the same connection method.
- Seamless connection: while a client is moving from one access point to another, mobile usage of the network must be possible within the same operator's zone.
- Seamless roaming: connectivity is important for a client. Connection must not be interrupted (in payment terms) when a user switches from one operator to another. Moreover, while a client is roaming in a network, considerable delay causes problems.
- Anonymity: in order to provide basic privacy, service has to be provided without need of having user's unique identity. It could be possible to use

pseudonyms for getting service. The challenge here is that when it is needed by law enforcement, it must be possible to reveal the identities of the malicious and untrustworthy users

- Mutual Authentication: for preventing malicious use of network, both user and network should be mutually authenticated. Moreover, man-in-the-middle and replay attacks must be prevented as well.
- No ultimate trust to service-providing operators: mostly users are considered as potential cheaters by the operators, but even the operators can cheat their clients. Moreover, home and foreign operators can cheat each other in roaming settlements. For instance, a user of operator *A* uses the network of operator *B*, but in the settlement phase operator *B* requests more funds than it has served.
- Three way honesty: clients cannot deny that they did not take service. Home and foreign operators cannot claim that they provide service more than they actually provided.
- Preventing Double Spending: it must not be possible to use a payment entity twice. In this respect, a user can use its monetary entity twice, or two clients can try to use the same payment entity at the same time. Both situations must be prevented.
- Unlinkability: it must not be possible to relate connection sessions of the users with other connection sessions. Hence, higher level of privacy could be provided.

4.2. General Overview of the Proposed Scheme

In our proposed system for providing access to the network, every operator that are also known as *service providers,* must get public-private key pair at the beginning. Instead of direct communication with operators by giving identities, clients get connection cards with unique *Serial Number (SN)*, and an *Initial Value* (*IV*), that are sealed with the operator’s private key. These two situations lead us to employ a trusted third party as a *connection card issuer* (*CCI*). Hence, each connection card is signed by the *CCI*. In Section 4.4 we will see details of this connection card structure and how the operators' public-private key pairs are distributed.

By hashing *IV* in connection card *n* times using hash chains($h^n(IV)$); we generate n tokens[1]. For starting a new connection, client concatenates *SN* with $h^n(IV),$ and encrypts it with the public key of the card owner operator. This starts the authentication phase, which we will mention in Section 4.5. Moreover, clients are not obliged to use whole connection card in one session. It can be consumed in several sessions.

After client is validated by the operator, this operator sends $h^n(IV)$ to the *access point* (*AP)* that the client already sent request. A challenge-response protocol using *HMAC* starts in this phase. In the third part of the Section 4.6, we will give the details of this protocol.

When client and the AP have authenticated each other, while getting service, clients must submit small tokens that are generated from the *IV* in the connection card to the operator. In Section 4.7, we will give the details of the ongoing connection and the disconnection phase.

In real-life scenarios, there exists more than one operator; hence, roaming details should be mentioned as well. Getting service from a foreign operator, i.e. roaming, will be explained in Section 4.8.

For unexpected interruptions in sessions, servers must be informed periodically. Hence, update packets must be sent to server. Moreover, clients' disconnection process must be properly handled. Packet update and disconnection phase details will be given in Section 4.9.

In Section 4.10 we will give the details of how the seamless connection is provided when a client moves from one *AP* to another in the home operator's area.

[1] In the following sections we have used the word token, and hash value synonymously.

4.3. Network Entities and Assumptions

The network entities used in our scheme are shown in Figure 4.1. For simulation of our network, we have used Omnet++ [11], which is an object-oriented modular discrete event simulator. Wireless Mesh Network support in Omnet++ is in its experimental phase. Hence, we adapt our scheme to actual Wi-Fi network elements, with a flexible structure to make it compatible with WMNs.

(a) Mobile Client (b) Access Point (c) Gateway (d) Operator Server

Figure 4.1. Network Entities

Mesh Routers have access point (*AP*), gateway and bridging capabilities. Therefore, a mesh router could serve like an access point that just relays the packets, and participates in routing protocols. Moreover, they could provide connection of the mesh network with backhaul network as a gateway. Hence counterpart of access point in Figure 4.1.b and gateway in Figure 4.1.c is mesh router. The only difference between access point and mesh router is that mesh routers can communicate in wireless medium, but access points are all bounded to the wired system.

One important detail about *AP*s is they all belong to a specific service provider, which is handled by operator identity information. Moreover each *AP* has a service cost for the size of the packets that they have transferred in service to a specific client. Similar to GSM, this service cost could vary according to the operator; or unlike GSM, depending on quality of service (QoS), service cost differentiates. These service cost, and operator identity information could be handled in two methods:

- Service cost and operator identity could be included in beacons of the *AP*s. Clients store these information, and choose the *AP* automatically with respect to the *AP* selection criteria given below.

- In SSIDs of the *AP*s service cost and operator identity could be given. Clients choose manually which *AP* they will connect.

Mobile station in Figure 4.1.a refers to clients that are mentioned as connection card holders. Clients send connection requests to the service providers. For the client side, it is important how the serving operator is chosen. An *AP* can be selected with respect to one or a combination of these criteria:

1. *AP* with the lowest service cost
2. *AP* with the best transmission power
3. *AP* from the home operator
4. *AP* with best quality of service

For our simulations, *AP* with best transmission power is selected by the clients.

The last entity, operator server shown in Figure 4.1.d, has two main functions as follows:

- Storing required connection card information
- Responding to all the connection requests

Table 4.1 gives the symbols that will be used in our proposed payment scheme.

Table 4.1. Symbols used in payment scheme

$h(.)$	Hash function that generates 512 bit hash value
$h^n(.)$	Hash chain function, means hash *n* times
$A\mid\mid B$	A concatenates with B
PU	Public Key
PR	Private Key
$E_{PU}(x)$	Encryption of x with Public Key
$D_{PR}(x)$	Decryption of x with Private Key
$E_{PR}(x)$	Signing x with Private Key
$D_{PU}(x)$	Validate x with Public Key
$HMAC_K(x)$	Hash x with the key of K

4.4. Connection Card Issuer

In this section, we give details of how the connection cards are issued and stored by the clients. Moreover, we mention about the token creation and distribution of the connection card information to the operators.

4.4.1. Connection Cards

For getting service from an operator, every client should get a connection card which is issued by connection card issuer (*CCI*), and each card is issued for a certain operator. Connection cards include tokens. While clients are getting service from the operator, they make the payment by periodically submitting tokens to the operator. Each connection card has a 128 bit unique *Serial Number (SN)*, an *Operator ID* that shows identity of the token holder's service provider, 128 bit *Initial Value (IV)*, that will be used as a seed for creating tokens, and a *Hash Count* that shows how many tokens that the current card contains (on the other hand how many times that the *IV* will be hashed for issuing first token). For legal reasons to get a connection card, clients must give their identities to the trusted *CCI*. *CCIs* are trusted third parties such that they do not divulge the identities to any service provider; they only give the data to law enforcement units when requested.

Operator's 1024 bit public key (PK_{OID}) is also attached to the connection cards. The data that a connection card (*CC*) has are shown as follows:

$$CC = SN \parallel Operator\ ID(OID) \parallel Initial\ Value(IV) \parallel Hash\ Count \parallel PK_{OID} \quad (4.1)$$

CC is signed by the private key of *CCI*, which gives us a 1024 bit signature of the *CCI*. Signature algorithm, signature, and expiration date of the connection card is also attached to the connection card. All the information a connection card contains is listed in Table 4.2.

We assume that all clients obtain the public key of the *CCI*. Each time a client purchases a connection card, it retrieves the *CC* information from this card, and authenticates if it is from the trusted *CCI,* by verifying the signature over the card.

Table 4.2. Connection Card Details

Serial Number (*SN*)	128 bit number. Unique for each connection card
Operator ID (*OID*)	Operator Id of the card owner
Initial Value (*IV*)	128 bit Initial Value that will be hashed to create tokens
Hash Count (HC)	Token count that will be generated from the *IV*
Signature Algorithm	signature algorithm information that is used to create signature (i.e. PKCS #1 SHA-1 With RSA Encryption)
Expiration Date	Date when the card will lose its validity
PK_{OID}	1024 bit public key of the Operator
Signature	1024 bit signature of the *CCI* over all fields.

4.4.2. Creation of Tokens

Our payment scheme is built upon micro payments, which denoted as *Tokens* in a connection card in our solution. Tokens are created by hashing *IV n* times which forms a hash chain as $h^n(IV)$. While getting service, clients must submit these tokens to the operator. After a certain usage of the network, in order to continue to get service, clients send following token, $h^n(IV) == h(h^{n-1}(IV))$, which is created by hashing *IV n*-1 times. After all the tokens are consumed, users must get a new connection card. Connection card information is stored in the client's database. Each time a new token is generated, n value which shows the remaining token count is stored as *Remaining Token Count* (*RTC*) in the client database. The *Last Hash Value* (*LHV*) that has been derived by hashing *IV RTC* times is also stored in the database. *RTC* is equal to *Hash Count* when a new unused connection card is purchased.

Token information stored in the client database is given below.

$$Stored\ Token\ Info = SN\ \parallel OID \parallel IV \parallel Hash\ Count\ \parallel RTC\ \parallel LHV \qquad (4.2)$$

4.4.3. Distribution of the Connection Card Information to the Operators

All the connection card information is signed and distributed by *CCI* to the relevant service providers with respect to the *Operator ID*s of the cards. In order to prevent operators from creating tokens in a connection card, instead of *IV*, only *Last Hash Value* (*LHV*) of the tokens, which is provided by hashing *IV Hash Count* (HC) times, is provided to these operators. Hence, this prevents operators from cheating other operators or the clients.

For storing connection card information, operators need a database. Track of a connection card in the operator database is as follows:

$$Stored\ Token\ Info = SN\ \parallel HC \parallel RTC\ \parallel LHV \parallel InUse \parallel Timeout \qquad (4.3)$$

As mentioned before, for an unused connection card *Remaining Token Count* (*RTC)* is equal to *HC*, and *LHV* is received from the *CCI*. During a session at which a client actively uses network, each time this client sends Hash values (tokens) to the operator for payment purposes, operator hashes each received hash value one time, and checks if it is equal to the stored *LHV*. At the end of session if it is validated, *RHC* is decremented by 1. When this value reaches to 0, the connection card becomes invalid. Details of this will be given in Section 4.5.

To prevent double use of the hash chains, *In Use* field is set as true, while a client is getting service. By this way if the users are trying to use same connection card, only one of them will get service. In case of connection interrupts, or some other disconnection reasons, timeout field is used, to set the token as not in use in a predefined time period.

4.5. Authentication

In this section we will give authentication details of our proposed scheme. In the beginning we assume that every client has at least one valid connection card, which is previously explained in Section 4.4.1. The necessary information we have retrieved from this card is shown as:

$$CC = SN \parallel Operator\ ID\ (OID) \parallel Initial\ Value(IV) \parallel HashCount \parallel PK_{OID} \qquad (4.4)$$

As we have mentioned in Section 4.3, depending on its hardware each *AP* broadcasts beacons that includes its *Operator ID* and *Service Cost,* or shares this information in its SSID. Whenever a client receives this information, it stores them in its memory as:

$$APInfo = Status \parallel IP \parallel ServiceCost \parallel Operator\ ID \qquad (4.5)$$

Service cost is the length of the packets that a client can transfer before sending a new token. For example if it is 4096, client should send a new token after the transfer of 4MB packets. *Status* is status of the *AP* that indicates current authentication situation as: *Not authenticated*, *authenticated*, or *associated*.

We discuss Authentication in three subsections:

1- Initial authorization of the client by service provider
2- Mutual authentication of *AP* and client (Challenge-Response protocol)
3- Authorization of a client in foreign domain

4.5.1. Initial Authorization

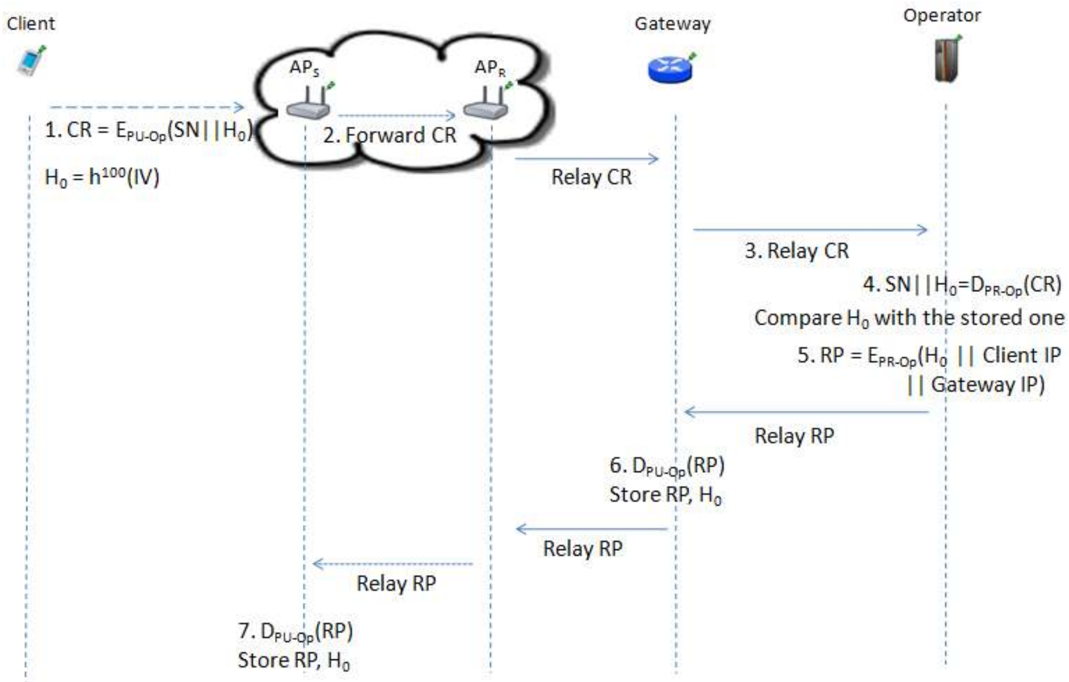
Client
AP_S
AP_R
Gateway
Operator
1. CR = $E_{PU\text{-}Op}(SN || H_0)$
$H_0 = h^{100}(IV)$
2. Forward CR
Relay CR
3. Relay CR
4. $SN || H_0 = D_{PR\text{-}Op}(CR)$
Compare H_0 with the stored one
5. RP = $E_{PR\text{-}Op}(H_0$ || Client IP || Gateway IP)
Relay RP
6. $D_{PU\text{-}Op}(RP)$
Store RP, H_0
Relay RP
Relay RP
7. $D_{PU\text{-}Op}(RP)$
Store RP, H_0

Figure 4.2. Initial Authorization

In Figure 4.2, connection medium between client and *AP* and between the *AP*s is wireless. Other than that all the lines are wired. Wireless communication is shown with dotted lines, and wired communication is shown with straight lines. As we have mentioned in Section 4.3, client chooses the *AP* with best transmission power. Chosen AP is denoted as AP_S (Serving Access Point) in Figure 4.2. Moreover, AP_R (Relaying Access Point) is the AP that is connected to the Internet backbone via gateway. In this figure, these two APs are a part of mesh network topology. This topology includes different number of APs and it is shown as a cloud. In this cloud, AP_S forwards the packets to AP_R through the other included APs in a multi-hop manner. This communication is shown by denser dotted lines.

Assume hash count of the first connection card that the client holds is 100, all the wired lines and the communication within the mesh network are encrypted, and connection card was never used before. Authentication steps are described below.

1. Client forms a connection request (*CR*) as follows:

(a) Client hashes the *Initial Value* (*IV*) *Hash Count* times:

$$H_0 = h^{100}(IV) \tag{4.6}$$

(b) Client concatenates *Serial Number* (*SN*) with H_0

(c) Client encrypts this value with the public key of the operator (*PU-Op*) which is included in the connection card.

$$CR = E_{PU-Op}(SN||H_0) \tag{4.7}$$

(d) Client sends *CR* to *AP*S

2. *AP*S creates an entry for the client in the database:

$$Client\ Entry = IP\ \parallel Status \parallel HashValue \parallel Decrement\ Amount \parallel PacketExceed \parallel Token\ In\ Use\ \parallel Timeout \tag{4.8}$$

IP refers to the IP address of the client. For the initialization phase, *Status* is set as *not authenticated*. *HashValue* is *null* at this point, and we will receive it from home operator in the reply packet. *Decrement Amount* and *Packet Exceed* will be explained in Section 4.6 (Packet Transfer). *Token in Use* field is for preventing double use, and DoS attack. It is set as true until the end of *Timeout* period or a disconnection request is received. *AP*S routes *CR* to its gateway, through relaying access points (*AP*R) in the mesh network.

3. Gateway creates client info in its database:

$$Client\ Entry = IP \parallel Hash\ Value \parallel Status \parallel Last\ AP\ IP \parallel Token\ in\ Use \tag{4.9}$$

All the information is same with the *AP*S, except the *Last AP IP* field. This is for routing reply packet from operator to the serving AP. Moreover, it is used for a seamless connection while a client is moving to another *AP*s coverage area. After storing the necessary information, gateway relays packet to the operator.

4. After the reception of *CR,* operator:

(a) Decrypts *CR* with its private key (PR-OP):

$$D_{PR-Op}(CR) = SN\ \parallel H_0; \tag{4.10}$$

(b) Checks whether the card info exists in the database;

(c) If it is not in use, compares received *Hash Value* with the stored one: $H_0 == LHV$. If it is valid, then the client is authenticated.

5. In order to form *Reply Packet* (*RP*) the operator:

(a) Signs $H_0 \parallel Client\ IP \parallel GatewayIP$ with its private key as follows:

$$RP = E_{PR-Op}(H_0 \parallel Client\ IP\ \parallel GatewayIP) \tag{4.11}$$

(b) Sends *RP* to the gateway with *GatewayIP*.

6. Gateway with the GatewayIP:
   (a) Validates the signature of the operator;
   (b) Updates *Hash Value* of the *Client Entry* with *Client IP* with H_0, and sets *Status* as Authenticated;
   (c) Relays *RP* to the AP_S mentioned in the *Client Entry* in the database. In the mesh topology, packet is first received by AP_R. After reception of the packet, it is routed to AP_S through the other APs in the mesh network.
7. AP_S performs the same operations with gateway. Hence, AP_S:
   (a) Validates the signature of the operator;
   (b) Updates *Hash Value* of the *Client Entry* with *Client IP* with H_0, and sets *Status* as authenticated;
   (c) Sends challenge request to client. (This will be explained in Section 4.5.3)

4.5.2. Reuse of a Connection Card

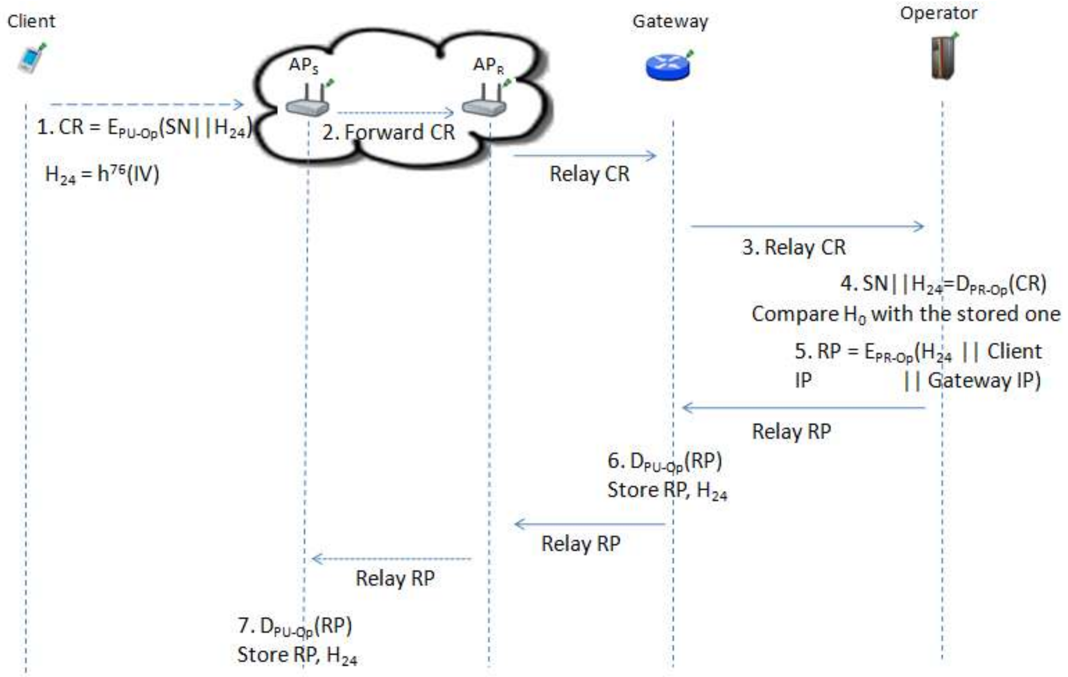

Figure 4.3. Reuse of a Connection Card

Reuse of the connection cards does not differ much from initializing a connection with a new connection card mentioned in previous section. Only difference is instead of sending first token H_0, one of the other tokens is sent as shown in Figure 4.3. Except that the other challenge is, *Stored Hash Value* in the operator's database must be properly updated.

Reuse of a connection card is described below.

1. Client forms a connection request (*CR*) as follows:
   (a) Client hashes the *IV Hash Count* times:
   $$H_{24} = h^{76}(IV) \tag{4.12}$$
   (b) Client concatenates *SN* with H_{24}
   (c) Client encrypts this value with *PU-Op*, which is included in the connection card.
   $$CR = E_{PU-Op}(SN \parallel H_{24}) \tag{4.13}$$

    (d) Client sends *CR* to AP_S

2. AP_S creates an entry for the client in the database. AP_S routes *CR* to its gateway, through relaying access points (AP_R) in the mesh topology.
3. Gateway creates client info in its database. Gateway relays *CR* to the operator.
4. After the reception of *CR,* operator:
    (d) Decrypts *CR* with its private key (*PR-OP*):

    $D_{PR-Op}(CR) = SN \; \| \; H_{24};$ (4.14)

    (e) Checks whether the card info exists in the database;

    (f) If it is not in use, compares received *Hash Value* with the stored one: $H_{24} == LHV$. If it is valid, then the client is authenticated.
5. In order to form *Reply Packet* (*RP*) the operator:
    (a) Signs $H_{24} \; \| \; Client \; IP \; \| \; GatewayIP$ with its private key as follows:

    $RP = \; E_{PR-Op}(H_{24} \; \| \; Client \; IP \; \| \; GatewayIP)$ (4.15)

    (b) Sends *RP* to the gateway with GatewayIP.
6. Gateway with the *GatewayIP*:
    (a) Validates the signature of the operator;

    (b) Updates *Hash Value* of the *Client Entry* with *Client IP* with H_{24}, and sets *Status* as authenticated;

    (c) Relays *RP* to the relevant *AP* mentioned in the *Client Entry* in the database. In the mesh topology, packet is first received by AP that is directly connected to the gateway. After reception of the packet, it is routed to AP_S through the other APs in the mesh network.
7. AP_S performs the same operations with gateway. Hence, AP_S:
    (a) Validates the signature of the operator;

    (b) Updates *Hash Value* of the *Client Entry* with *Client IP* with H_{24}, and sets *Status* as authenticated;

    (c) Sends challenge request to client.

4.5.3. Access Point Authentication

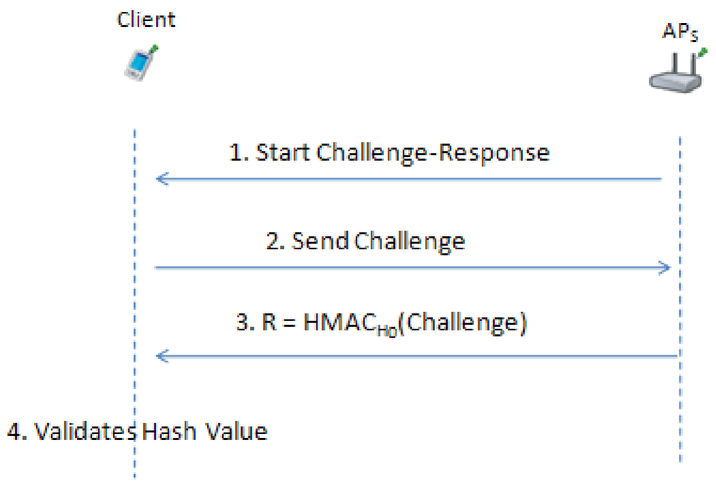

Figure 4.4. Access Point Authentication Using a Challenge-Response Protocol

After validation of Reply Packet received from operator, AP_S starts a challenge-response protocol for mutual authentication between client and AP_S.

As shown in Figure 4.4, AP_S authentication is described below.

1. AP_S sends a challenge request to the client which started connection.
2. When client receives this challenge request, it sends a 128 bit challenge to the AP_S
   (a) Client drops the packet if it is not the *AP* that he sent connection request.
   (b) Client drops the packet if there was not any authentication request.
3. AP_S hashes this challenge, and uses relevant *Hash Value* (here H_0) which is stored as *Last Hash Value* (*LHV*) as the key of HMAC:
   $$R = HMAC_{LHV}(Challenge) \quad (4.16)$$
   AP_S sends response *R* to the client.
4. Client also hashes the challenge and uses the stored *Hash Value* (H_0) as the key of HMAC. Then compares $HMAC_{HV}(Challenge) == HMAC_{LHV}(Challenge)$. If it is authenticated client sets the AP_S as *authenticated*.
5. Instead of adding a new step to the challenge-response protocol, as soon as AP_S gets the next token from the client, it sets the *Status* of the client as *associated*.

4.6. Packet Transfer

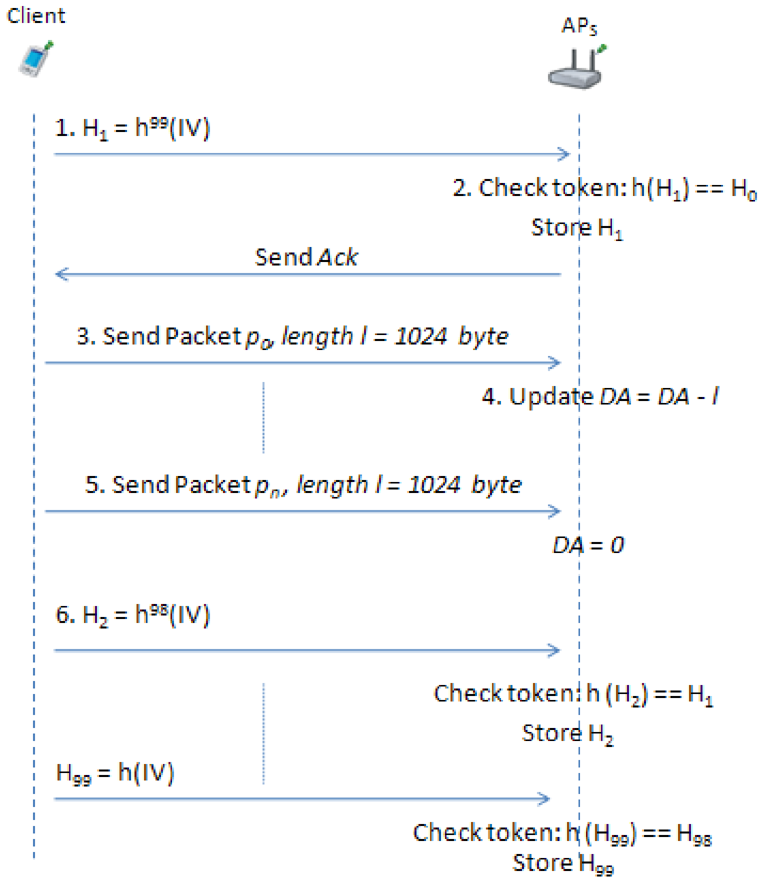
Client
AP_S
1. $H_1 = h^{99}(IV)$
2. Check token: $h(H_1) == H_0$
Store H_1
Send *Ack*
3. Send Packet p_0, *length l = 1024 byte*
4. Update *DA = DA - l*
5. Send Packet p_n, *length l = 1024 byte*
DA = 0
6. $H_2 = h^{98}(IV)$
Check token: $h\ (H_2) == H_1$
Store H_2
$H_{99} = h(IV)$
Check token: $h\ (H_{99}) == H_{98}$
Store H_{99}

Figure 4.5. Packet Transfer

After Mutual authentication of client and AP_S, client starts to send packets as shown in Figure 4.5.

1. Before sending data packet, client sends next token (H_1) first. (It causes client to spend one token for each connection session.)
2. AP_S gets H_0 from the *Client Entry* in the database. Then:
   (a) Checks if $h(H_1) == H_0$
   (b) If true sends acknowledgement (*Ack*) to *Client*, and sets *Client* as *associated*
3. Client sends first 1024 byte data packet p_0 .

4. *AP*$_S$ decrease *Decrement Amount* (*DA*) with the length of the packet (here 1024 byte). When client is first authenticated *DA* was set as the *Service Cost* of the *AP*$_S$.
5. *DA* value is also stored in client's database. Therefore, client proceeds sending data packets until *DA* value drops to 0.
6. Client sends next hash value (H_2).
   (a) If *DA* goes below 0, with every received packet *AP*$_S$ increments *Packet Exceed* field by packet length. Hence, client could proceed sending packets for a while.
   (b) When H_2 is received and validated, *DA* is set to *Service Cost* of the *AP*$_S$ and *Packet Exceed* value is decremented from *DA*. Then *Packet Exceed* set to 0.
   (c) If *Packet Exceed* value exceeds a predefined limit. Client is set as *Not Authenticated,* and all the packets received from this client will be ignored.
7. Every time *DA* goes below 0, client sends next token.

4.7. Roaming

If the operator of the connection card is different from the home operator, then client will roam between these two service providers. According to Figure 4.6 steps are as follows:

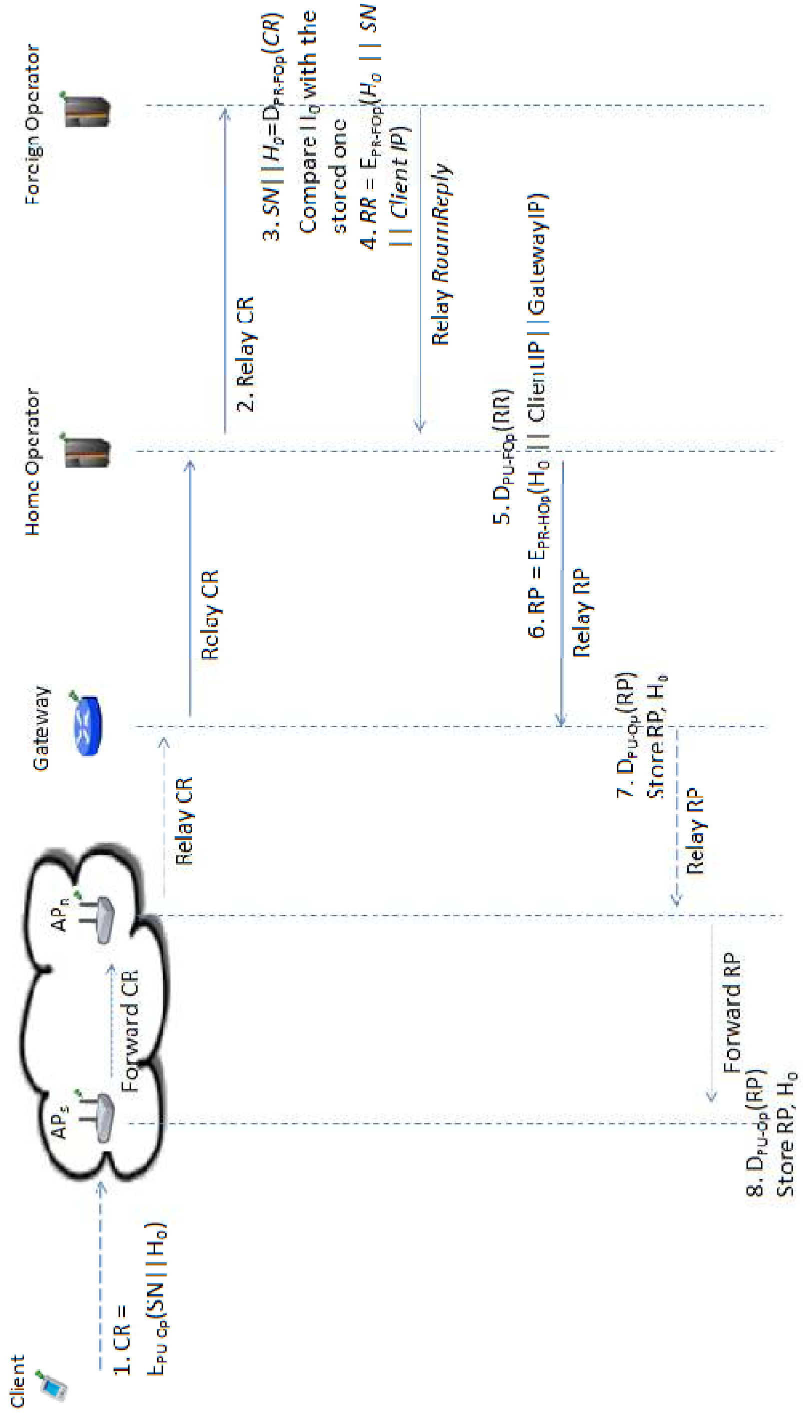

Figure 4.6. Roaming

1. Client forms a connection request as mentioned in Section 4.3.1, Step 1.
   (a) $SN \parallel H_0$ is encrypted with the public key of foreign operator (PU-FOp):

   $$CR = E_{PU-FOp}(SN \parallel H_0) \tag{4.17}$$

   (b) AP_S creates *Client Entry* in its database as in authentication, Step 2.

   (c) AP_S routes CR to AP_R within the mesh network.

(d) AP_R relays CR to gateway.

(e) Gateway creates *Client Entry* in its database as in authentication, Step 3.

(f) Gateway relays CR to home operator.

2. Home operator creates a roaming token entry (*RTE*).

(a) $RTE = ClientIP \parallel HashValue \parallel SN \parallel TotalHashCount \parallel inUse$. (4.18)

Sets *Hash Value* as "waiting" and *inUse* field as false.

(b) Home operator relays *CR* to foreign operator.

3. After the reception of *CR,* foreign operator:

(a) Decrypts *CR* with its Private Key (PR-FOp):

$D_{PR-FOp}(CR) = SN \parallel H_0$; (4.19)

(b) Checks whether the card info exists in the database;

(c) If it is not in use, compares received hash value with the stored one: $H_0 == LHV$. If it is valid, then the client is authenticated.

4. Foreign operator forms roaming reply packet (*RR*) and signs this packet with its private key:

(a) $RR = E_{PR-FOp}(H_0 \parallel SN \parallel Client\ IP)$ (4.20)

(b) Then it relays this packet to the home operator.

5. When home operator receives *RR*

(a) It verifies the signature with the public key of foreign operator:

$D_{PU-FOp}(RR)$. (4.21)

(b) Stores H_0 and *RR* in its database.

6. Home operator forms a new reply packet:

(a) Finds the *GatewayIP* information from the database with respect to the *ClientIP*

(b) Signs $H_0 \parallel Client\ IP \parallel GatewayIP$ with its private key:

$RP = E_{PR-Op}(H_0 \parallel Client\ IP \parallel GatewayIP)$ (4.22)

(c) Relays *RP* to the gateway with *GatewayIP*

7. Gateway with the GatewayIP:

(a) Validates the signature of the operator;

(b) Updates hash value of the *Client Entry* with *Client IP* with H_0, and sets *Status* as Authenticated;

(c) Relays *RP* to the relevant *AP* mentioned in the *Client Entry* in the database. In the mesh topology, packet is first received by AP that is directly connected to the gateway, *AP_R*. After reception of the packet, it is routed to *AP_S* through the other APs in the mesh network.

8. *AP_S* performs the same operations with gateway. Hence *AP_S*:
   (a) Validates the signature of the operator;
   (b) Updates hash value of the *Client Entry* with *Client IP* with H_0, and sets *Status* as authenticated;
   (c) Sends challenge request to client for the challenge-response protocol explained in Section 4.5.3.

4.8. Update Packets and Disconnection

For connection interruptions, *AP*s and gateways send periodically update packets. Moreover, for reusing connection cards, clients must disconnect properly from the system. In this section we give details of update packets, as disconnection phase.

4.8.1. Update Packets

As we have mentioned in Section 4.6, after the authentication phase, validation of the micropayments are handled by the *AP*s, for decreasing the burden on the operator server. Even in this situation servers must be informed periodically to prevent any problem when there is an interruption in the connection.

Sending update packets is performed in two steps:

1. *AP*s send update packets to gateways
2. Gateways send update packets to operator servers

For supporting a seamless mobile communication in home operator, if the mobile client, which is moving from one *AP*s zone to another, is validated in gateway layer, this could lower the authentication delay in considerable amounts. The reason is server is not interrupted, and public key encryption steps are bypassed. The motivation for sending update messages to gateway first, instead of sending immediately to server, lies behind this.

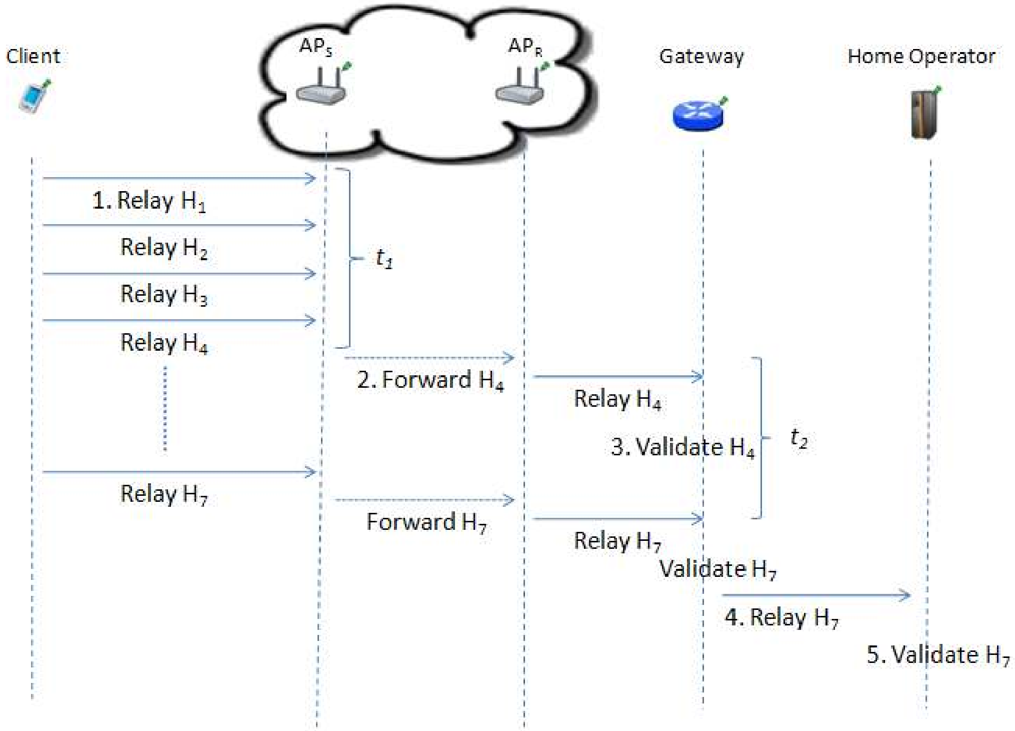

Figure 4.7. Update Packets

In Figure 4.7 all the wired lines are encrypted. With respect to this figure, update packets are evaluated as follows:

1. When client sends first token after the connection is initialized, *AP* starts a time-out period of t_1 seconds for sending update messages to gateway.
2. AP_S routes last received and validated token (here H_4) to AP_R within the mesh network. With the reception of H_4, AP_R relays it to gateway.

3. Gateway hashes the received token, until it is equal to the stored *Hash Value*. If it is validated then hash value in the database is updated.
4. After t_2 seconds passed till the last Update Packet received from AP_S, gateway sends last received and validated token (here H_7) to home operator.
5. Operator hashes the received token. Afterwards check if it is equal to the stored *Hash Value*. If it is validated then hash value in the database is updated and *Remaining Token Count* value is decreased by *1*.

4.8.2. Disconnection

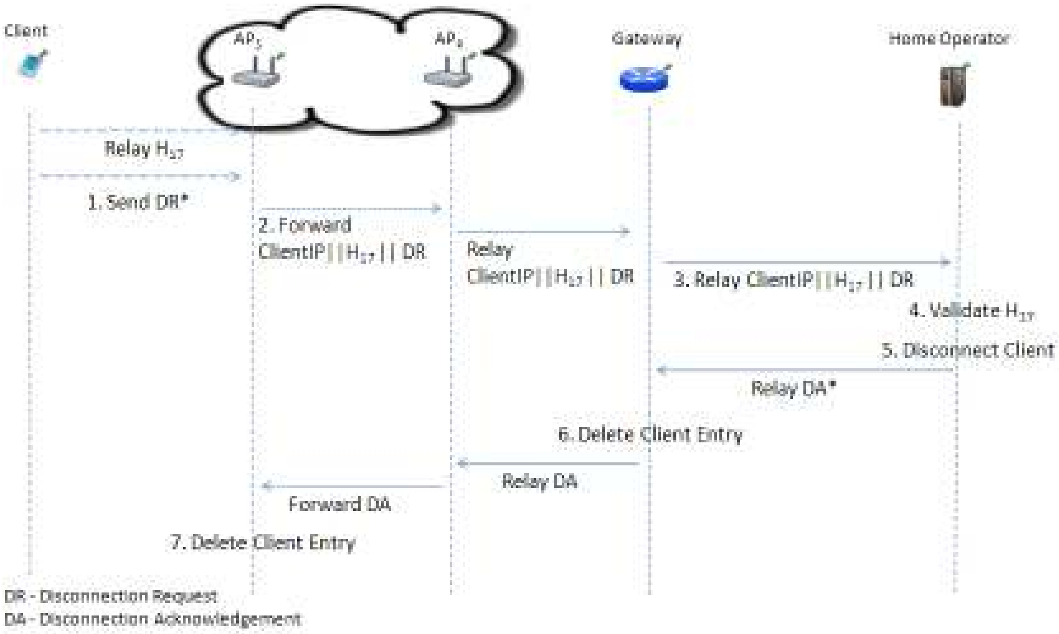

Figure 4.8. Disconnection (Home Operator)

For reusability of the connection cards, it is important for a client to properly disconnect from the system. There are two different disconnection cases: disconnection from home operator, and disconnection while roaming.

Disconnection from home operator is shown in Figure 4.8. Disconnection occurs as follows:

1. After client sends last used token to the *AP*$_{S}$*,* sends a disconnection request (*DR*).
2. *AP*$_{S}$ updates the *Client Entry Status* as *not authenticated*, and routes Last Token (here *H*$_{17}$) with *ClientIP*, and *DR* to *AP*$_{R}$ within the mesh network.
3. Gateway validates *H*$_{17}$ and updates the *Client Entry Status* as *not authenticated*. Afterwards gateway relays last token (here *H*$_{17}$) with *ClientIP*, and *DR* to gateway.
4. Operator hashes *H*$_{17}$, and checks whether if it is identical with stored *Hash Value*. If it is validated, then *Remaining Hash Value* is decremented by 1 and *Hash Value* is updated with *H*$_{17}$.
5. Operator sets *In Use* field of the *Client Entry* as false. Afterwards operator relays *disconnection acknowledgement* (*DA*) with the *Client IP* to gateway.
6. As soon as gateway receives *DA*, it deletes the related *Client Entry.* Then gateway routes *DA* to the AP that is directly connected to the gateway, *AP*$_{R}$. After reception of the packet, it is routed to *AP*$_{S}$ through the other APs in the mesh network.
7. *AP*$_{S}$ deletes *Client Entry* with the reception of *DA*, as well.

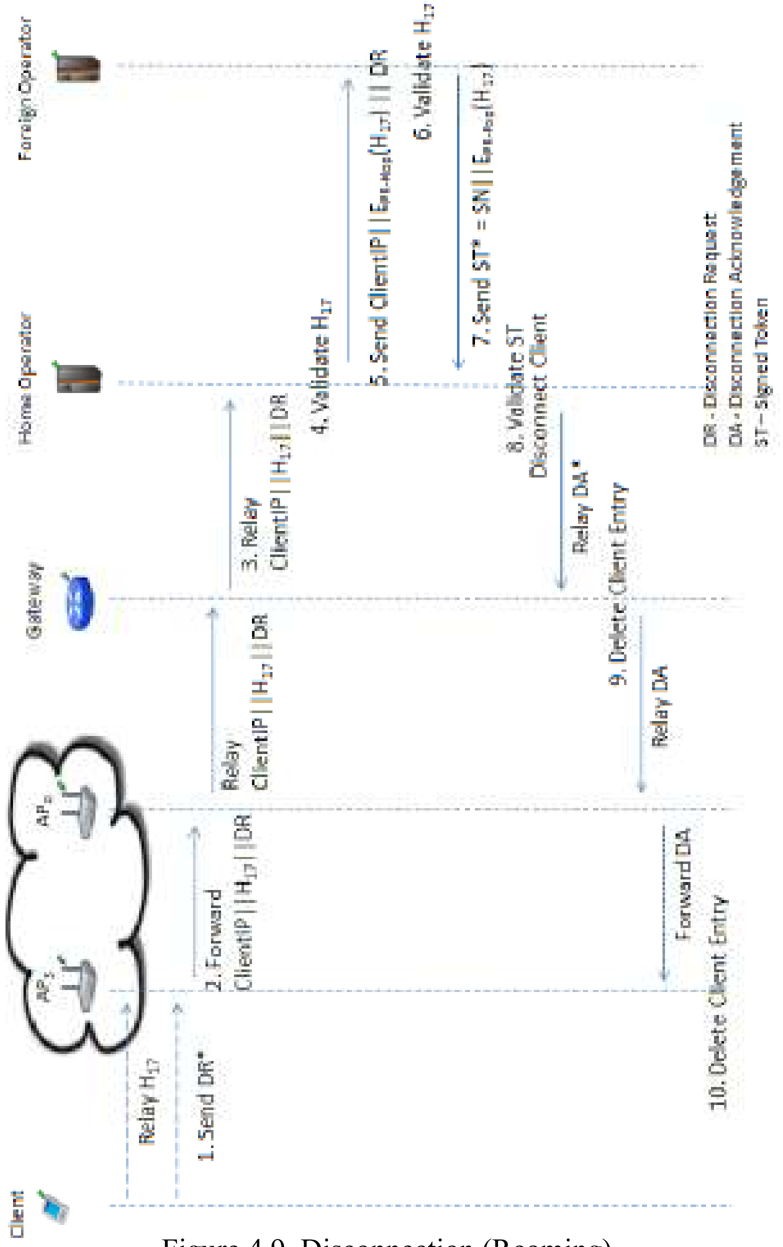
Client
AP_S
AP_R
Gateway
Home Operator
Foreign Operator
Relay H_{17}
1. Send DR*
2. Forward ClientIP || H_{17} || DR
Relay ClientIP || H_{17} || DR
3. Relay ClientIP || H_{17} || DR
4. Validate H_{17}
5. Send ClientIP || $E_{FO-HO}(H_{17})$ || DR
6. Validate H_{17}
7. Send ST* = SN || $E_{FO-HO}(H_{17})$
8. Validate ST Disconnect Client
Relay DA*
9. Delete Client Entry
Relay DA
Forward DA
10. Delete Client Entry
DR - Disconnection Request
DA - Disconnection Acknowledgement
ST – Signed Token

Figure 4.9. Disconnection (Roaming)

For the roaming, disconnection phase is shown in Figure 4.9. Disconnection details are given below.

1. After client sends last used token to the *AP*$_S$*,* she sends a *DR*.
2. *AP*$_S$ updates the *Client Entry Status* as *not authenticated*, and routes last token (here *H*$_{17}$) with *ClientIP*, and *DR* to *AP*$_R$ within the mesh network.

3. Gateway validates H_{17} and updates the *Client Entry Status* as not authenticated. Afterwards operator relays last token (here H_{17}) with *ClientIP*, and *DR* to gateway.
4. Home operator hashes H_{17}, and checks whether if it is identical with stored *Hash Value*. If it is validated, then *Remaining Hash Value* is decremented by 1. For settlement among the operators, home operator stores SN and H_{17}.
5. Home operator signs Last Token, here H_{17}. Afterwards, operator relays last token with *Serial Number* of the connection card, and *DR* to foreign operator.
6. Foreign operator hashes token until it is identical with stored *Hash Value*. If it is validated, then *Remaining Hash Value* is decremented and *Hash Value* is updated. For this example, if the stored *Hash Value* is H_0, H_{17} is hashed 17 times. Afterwards, *Remaining Hash Value* is decremented by 17 and *Hash Value* is updated with H_{17}. For the settlement phase, signed last token is stored.
7. Foreign operator sets *In Use* field of the *Client Entry* as false. It forms *Signed Token* (*ST*) by signing SN || H_{17}. Sends *ST* to home operator.
8. Operator validates *ST*. If it is validated, it stores *ST* for the settlement phase. Operator sets *In Use* field of the *Client Entry* as false. Afterwards operator relays *Disconnection Acknowledgement* (*DA*) with the *Client IP* to gateway
9. As soon as gateway receives *DA*, it deletes the related *Client Entry.* Then gateway routes *DA* to the AP that is directly connected to the gateway, AP_R. After reception of the packet, it is routed to AP_S through the other APs in the mesh network.
10. AP_S deletes *Client Entry* with the reception of *DA*, as well.

4.9. Seamless Mobility in Home Operator

When a client moves out of the coverage area of associated AP, then a new connection initialization process should be started. If the client is still in the same operator’s zone, it is possible not to start the authentication phase from scratch.

In this setting, if a client moves from one AP's zone to another, it just passes the previously recorded information to the new AP. Each AP has a key pair and registered to a specific gateway. Gateways are responsible from a local zone. Moreover, we assume that gateways are trusted third parties and they broadcast public keys of the registered APs to the other registered APs within their local zone. Here distribution of the key pairs is out of our scope of the thesis.

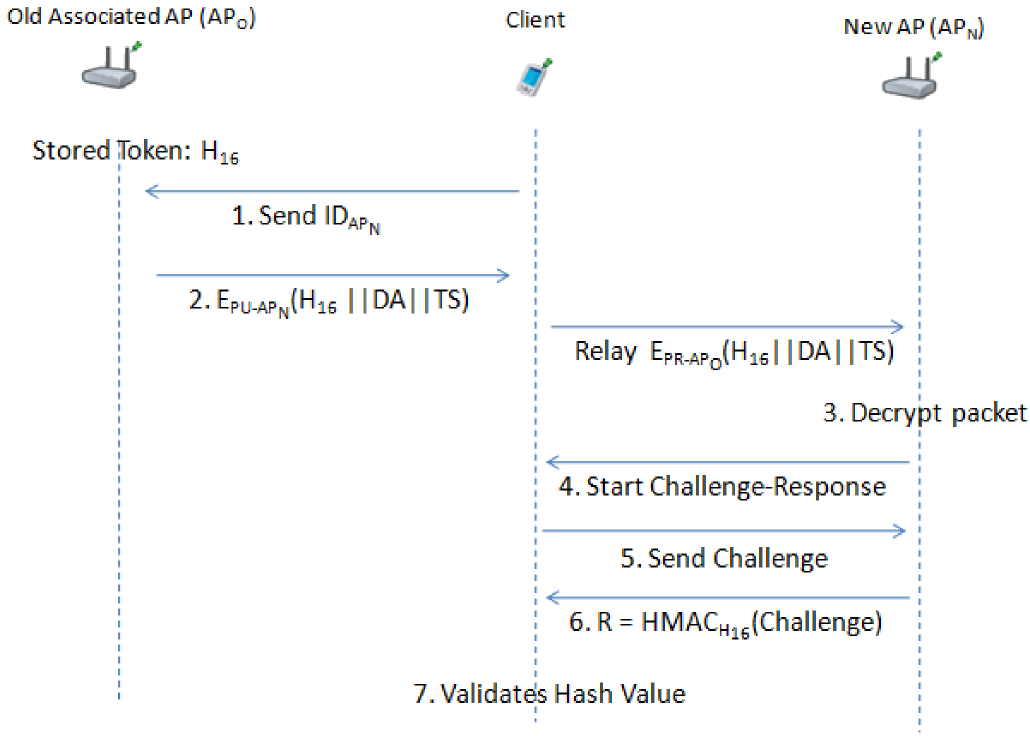
Old Associated AP (AP_O)
Client
New AP (AP_N)
Stored Token: H_{16}
1. Send ID_{AP_N}
2. $E_{PU\text{-}AP_N}(H_{16} || DA || TS)$
Relay $E_{PR\text{-}AP_O}(H_{16} || DA || TS)$
3. Decrypt packet
4. Start Challenge-Response
5. Send Challenge
6. $R = HMAC_{H_{16}}(Challenge)$
7. Validates Hash Value

Figure 4.10. Mobile Client

In Figure 4.10 client moves to the new AP's (AP_N) coverage area from old AP (AP_O). In this figure last used token is taken as H_{16}, and remaining length of data packet is shown as *DA*. Seamless mobility is provided as follows:

1. Client sends identity information of the new AP (AP_N) to AP_O.
2. AP_O concatenates H_{16}, *DA*, timestamp *TS*, and encrypts it with the public key of AP_N. It relays this packet to AP_N via client.
3. AP_N decrypts packet and checks whether it is expired.

4. If it is not expired, AP_{N} stores H_{16} and DA. Then AP_{N} sends a challenge request to the client.
5. When client receives this challenge request, it sends a challenge to the AP_{N}
6. AP_{N} hashes this challenge, and uses relevant *Hash Value* (here H_{16}) as the key of HMAC:

$$R = HMAC_{LHV}(Challenge) \tag{4.23}$$

AP_{N} sends response R to the client.

7. Client also hashes the challenge and uses H_{16} as the key of HMAC. Then client compares it with the received R packet. If it is authenticated, client sets the AP_{N} as the new authenticated AP.

4.10. Settlement among the Operators

As we have mentioned in Section 4.7, clients are capable of roaming between two operators. From time to time the operators should run settlement procedure in order to transfer funds among themselves for the services provided to other operators' customers.

Important issue for settlement is that, all the used tokens must be stored in both *Serving Operator*'s (O_{S}) and *Token Owner Operator*'s (O_{TO}) database. For every established connection, O_{S} stores token usage logs as follows:

$$Roaming\ Token\ Log = SN \parallel OID \parallel\ Signed\ First\ Token\ \parallel Signed\ Last\ Token \tag{4.24}$$

For every roaming session, logs are stored in O_{TO}'s database as follows:

$$Token\ Log = SN \parallel Serving\ Operator\ ID \parallel Signed\ First\ Token \parallel Signed\ Last\ Token \tag{4.25}$$

Signed First Token is received in the authorization phase from the foreign operator. Moreover, *Signed Last Token* is received in the disconnection phase from the foreign operator as well. In the settlement among the operators, O_S sends first and last used token information to the O_{TO}. Current usage of the client is derived by hashing last used token until we get first used one. For instance, if first used token is H_{24} and last used token is H_{30}. O_{TO} hashes H_{30} 6 times, and it has to make payment to O_S for used 6 tokens.

If O_S uses consecutive tokens of a connection card only one time, it does not cause any problems. The problem arises when different parts of a connection card are used by the same O_S.

For instance suppose a connection card CC_0 includes 100 tokens. A client that wields CC_0 has spent:

- Token H_0 to H_{15} in O_S*'s* zone (First connection session);
- H_{16} to H_{22} in O_{TO}'s zone (Second connection session);
- H_{23} to H_{29} in O_S's zone again (Third connection session).

It is very easy for O_S to obtain all the tokens from H_0 to H_{29} by hashing H_{29} 29 times. Hence, O_S can cheat the O_{TO} by saying that it has provided all the service for the client in exchange of the derived tokens. To prevent this, when O_{TO} sends first token to O_S, it signs and send it back to O_{TO}. Moreover, when a roaming client sends a disconnection request, O_S signs last used token received from the client. Afterwards, it sends this *Signed Last Token* to $O_{TO.}$

For this example, in first connection session, O_{TO} receives first and last used tokens that are signed by O_S, $E_{PR\text{-}Os}(H_0)$, $E_{PR\text{-}Os}(H_{15})$. Moreover, in third connection session, O_{TO} receives first and last used tokens that are signed by $O_{S,}$ $E_{PR\text{-}Os}(H_{23})$ and $E_{PR\text{-}Os}(H_{29})$. If O_S claims that it has given all the service to the client with these received tokens, O_{TO} can prove that O_S only used tokens from H_0 to H_{15}, and tokens from H_{23} to $H_{29.}$ If O_S does not send signed tokens it cannot be paid for the service it has given.

For the abovementioned example, there exists another problem. After O_{TO} had received H_{22}, it can claim that it has given all the service for tokens from H_0 to $H_{22,}$ instead H_{16} - H_{22}. Therefore, for every connection and disconnection request O_{TO} signs the tokens.

For this example, in first connection session, O_S receives first and last used tokens that are signed by O_{TO}, $E_{PR\text{-}O_{TO}}(H_0)$, $E_{PR\text{-}O_{TO}}$ (H_{15}). Moreover, in third connection session, O_S receives first and last used tokens that are signed by O_{TO}, $E_{PR\text{-}O_{TO}}(H_{23})$ and $E_{PR\text{-}O_{TO}}(H_{29})$. If O_{TO} claims that it has given all the service for the tokens from H_0 to $H_{22,}$ O_S can deny it with the received tokens $E_{PR\text{-}O_{TO}}(H_0)$, $E_{PR\text{-}O_{TO}}$ (H_{15}).

The only problem that the operators could encounter here rises if one of the operators does not send the first or last signed token. In initial authorization phase, if O_S does not send first signed token, it would not be paid by O_T for the provided service. Moreover, if O_T cheats O_S and does not send first signed token, O_S does not give service to the client. For disconnection phase, if O_S does not send the first signed token to O_T, then O_T does not give service as well. If signed last token is not received by O_T, O_S will not be paid for the service. On the other hand if O_T does not send last signed token to O_S, O_S will not provide service to O_T's clients in exchange of the funds. Hence, cheating among the operators is prevented.

4.11. Discussion

In this section, we discuss which of the requirements given in Section 4.1 are met.

Roaming/mobility: As we have seen in Section 4.5.2, reuse of a connection card is possible after attempting first connection. Roaming is supported, when our protocol is implemented in participating *AP*s, and tokens are valid.

Seamless connection: As shown in section 4.9, mobility of the users in home operator is supported. Hence, clients in the same operator can move from one *AP* to another without any interruptions in their connections.

Seamless roaming: Mobility of the clients from one operator's zone to another is not provided without connection interruptions. This requirement is not met yet, but our infrastructure supports it.

Anonymity: As mentioned in Section 4.4.1, for legal purposes users must give their identities to connection card issuer (*CCI*) for getting connection cards. Therefore, as far as *CCI* keeps clients' identities secret, users can stay anonymous.

Mutual authentication: In Section 4.5.1 we have seen how the client is authenticated by the server. In Section 4.5.3, valid token information is received by the *AP*, and with the challenge-response protocol both AP and the Client is mutually authenticated.

If there is an adversary between AP and the Client that intercepts the packet transfer between these two entities, in initialization phase, he can behave like the client. After the authentication phase, the adversary gets service from the Operator. Without getting service, client does not send the next token. Hence, client only loses two tokens in this situation; first is for establishing connection, second is for packet transfer.

If the client is already authenticated, and while sending next token if the packet is captured by the adversary, because of the lack of the Serial Number knowledge, it is not usable by him.

No ultimate trust to operators: In our scheme, users control their balance in the connection cards. Operators cannot generate tokens and it is not possible for the operators to retain unused tokens. Hence, they cannot cheat the users by saying "the token is already used".

Three-way honesty: Since the tokens are issued by *CCI*, only the *CCI* and connection card holder knows all the tokens that are related with a specific connection card. Hence whenever a Client sends a new token, it is not possible for him to say "I did not use it". Since *CCI* is a trusted third party, in the roaming phase, operators cannot say that they provided service for non-used tokens.

Preventing double spending: All the connection card information is stored in the database with *In Use* fields. Therefore it is not possible for two users to use the same connection card at the same time. Since the last token information is stored in the database, it is not possible to double-spend a token.

Unlinkability: Our protocol does not provide unlinkability yet.

5. SIMULATION DETAILS AND PERFORMANCE EVALUATION

In previous section, we have given the design details of our prepaid payment scheme. In this section, we will give how the protocol is simulated, simulation environment and network entities, delays of the cryptographic operations, network structure and performance evaluation.

For performance evaluations we have implemented our protocol using Omnet++ [11], which is an object-oriented modular discrete event simulator. Omnet++ only provides infrastructure, hence it is supported by other open source frameworks such as INET Framework [12], Mixim [13], Oversim [14] etc.

For the simulation of wireless networks, we have developed our protocol over INET framework. Mesh Network structure is still in experimental progress in this framework. Therefore, we have implemented our protocol over current IEEE 802.11g Wi-Fi standard.

With the assistance of modular structure of Omnet++, we try not to modify anything in the link layer; hence we have implemented our payment module in the application layer. Therefore all the TCP and UDP packets are interpreted in our payment module before they are sent to/received from network layer.

5.1. Simulation Details

In this section, we give the simulation details of our protocol.

5.1.1. Environmental Details

Clustered network structure supported in [9] is adapted to our simulations. This is shown in Figure 5.1. All the *AP*s are connected to gateways with 4 Mbps point-to-point (PPP) lines. Some important specifications about the *AP*s are shown in Table 5.1. Here *Service Cost* shows the packet transfer rate for each token that the client sends. *Update Interval* is the time for sending next update packet to gateway

Table 5.1. AP Specifications

AP-Gateway Connection bit rate	4 Mbps - PPP
AP Transmission bit rate	108 Mbps
AP-Gateway Distance	10 m (approximately)
Service Cost	8 MB
Update Interval	1 minute

Gateways are for connecting *AP*s to Internet backbone. Apart from the other gateways in the backbone, these store the client information for seamless connection. Gateways are connected each other via 1 Gbps PPP lines. For gateways, *Update Interval* is 2 minutes, and all the update packets are sent to home operator.

Our network Topology consists of 13 gateways as shown in Figure 5.2, 32 *AP*s as shown in Figure 5.3 and variable amount of mobile clients.

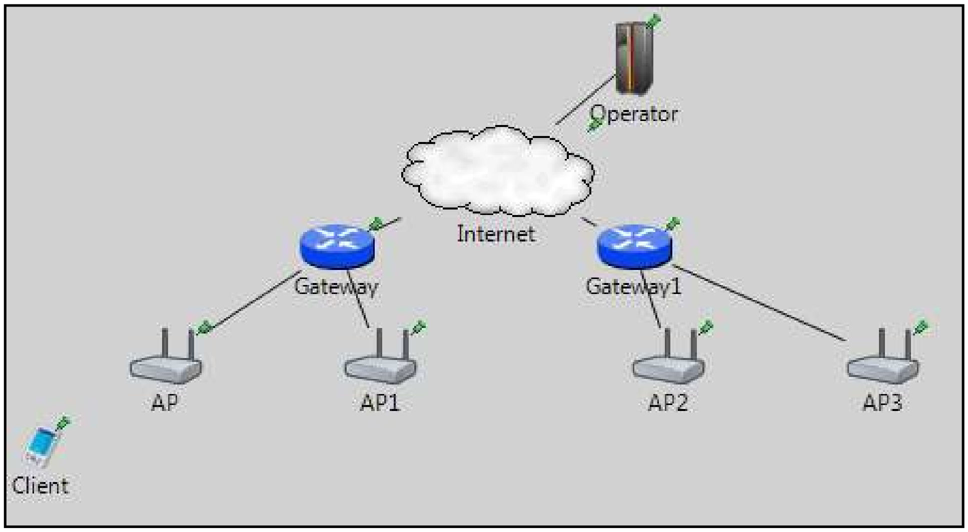

Figure 5.1. Clustered Structure

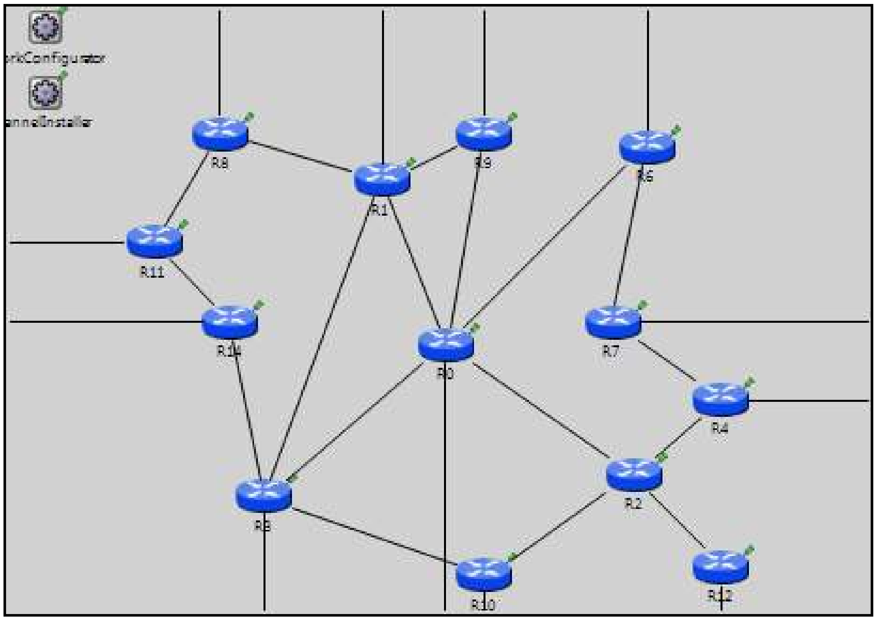

Figure 5.2. Gateway backbone

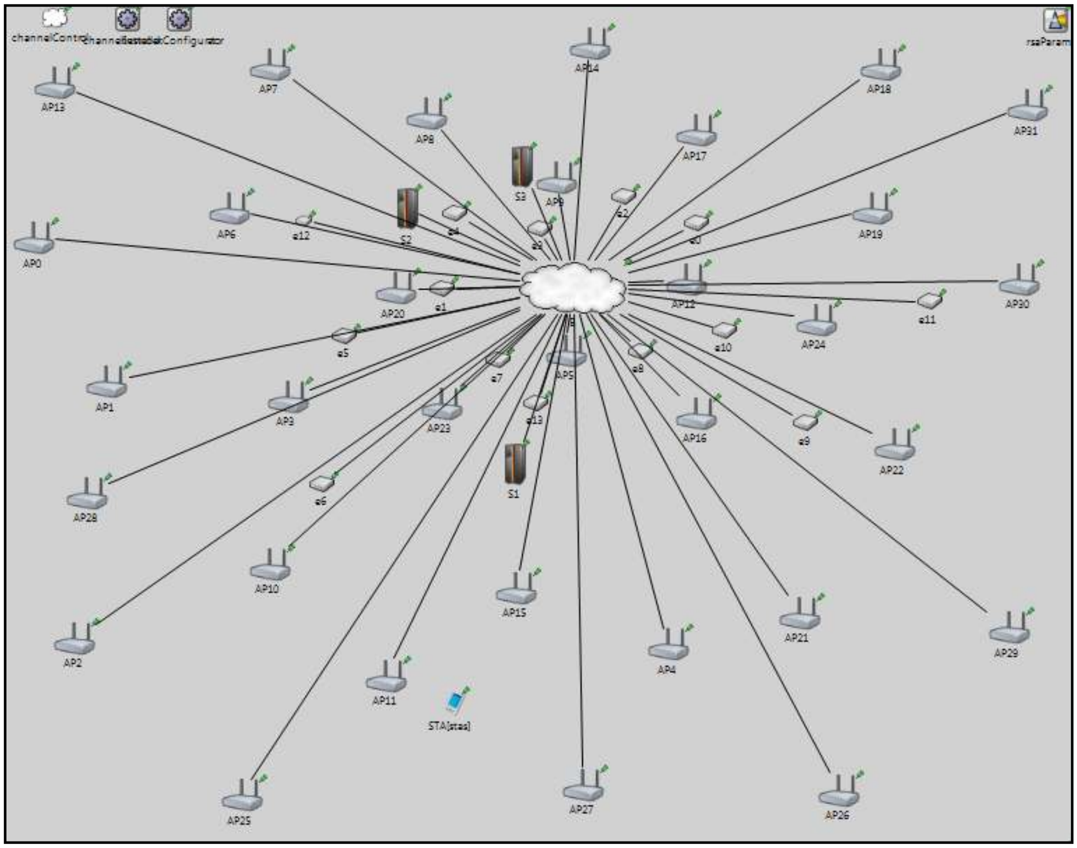

Figure 5.3. Simulation Environment (*AP*s and Servers)

5.1.2. Public Key Operations and Their Timings

Omnet++ does not support any cryptographic operations; therefore, we have integrated Crypto++ [15] to our simulations. However, it was not possible to get processing times of the encryption/decryption operations in a specific network device. Accordingly for gateways and *AP*s, we have used public key cryptography processing timings from [10]. For operator servers, we have used timings from [16], which gives us performance metrics of a dual-core 64-bit Intel Xeon 2.8 GHz processors. For mobile clients, we have used performance values from [27] which uses Intel Celeron D 351 3.2 GHz processor and encryption time is 31*ms*.

Platform specifications are shown in Table 5.2, and RSA timings are shown in Table 5.3.

Table 5.2. Platform Specifications

	Gateway [10]	Linksys WRT54GS (AP) [10]	Server [16]	Client [27]
CPU Speed	2.08 GHz	200 MHz	Dual-core 64 bit 2.8 GHz	3.2 GHz
CPU type	AMD Athlon XP 2800	Broadcom MIPS32	Intel Xeon	Celeron D 351
RAM	512 MB	32 MB	-	-

Table 5.3. 1024 bit RSA timings

	Gateway [10]	Linksys WRT54GS [10]	Server [16]	Client [17]
RSA Encryption	0.4 ms	12.3 ms	0.4 ms	31 ms
RSA Decryption	9.0 ms	300.6 ms	9.0 ms	-

5.1.3. Storage Requirements

In initial authorization section (Section 4.5.1.), we give how the client entries are stored in APs databases. In Table 5.4 we give storage requirements of each client entry.

Table 5.4. Storage requirements for each client

Client Entry	IP	Status	Hash Value	Decrement Amount	Packet Exceed	Token in Use	Timeout
Size (Bit)	32	2	512	32	32	1	16

As shown in Table 5.4, 627 bit of information is stored in the AP memory per connected client. Hence 632 bit memory allocation is needed for each client. In addition to this, each AP allocates 1024 bit for operator's public key. As given in burst scenario (Section 5.2.1), when there exists 600 clients in the simulation environment, maximum number of simultaneously connected client count in an AP is 28. This shows us for 28 clients, 28*632 bits = 2198 bytes of memory space is required. Hence an AP with 32 MB memory space is adequate for serving a considerable amount of clients and storing the other static values as public keys.

5.1.4. Performance Metrics

We use four performance metrics.

1. *End-to-end authentication latency* is the average time elapsed between sending the connection request (by client) and completion of authentication.
2. *Server service time* is the average duration for a server to complete the processing of a connection request packet. This metric also includes the waiting time of the message in the queue.
3. *Connected node count* is the number of simultaneously connected nodes to network.
4. *Connected node count per AP* gives the average and maximum number of nodes that are served by a single AP.

5.2. Simulation Scenarios and Results

In this section, we give simulation results. In the first section, we give a burst scenario, where clients send connection requests simultaneously. In the second section, we simulate a real life scenario, where the clients intermittently connect to the system and start TCP sessions. In the third section, we combined these two given scenarios as a rush scenario. In this scenario, while users are sending TCP packets, at a certain time, 200 clients send connection requests simultaneously. In the last section, we simulate a mobile real-life scenario with multiple operators. In this scenario, there exists three different operators and with every disconnection users move to a random point in the area.

5.2.1. Burst Scenario

Simulations in this section are for getting results how the infrastructure responds when all the connection requests are obtained at the same time. In this scenario all the uniformly distributed nodes send connection requests in first 5 seconds.

We examined this scenario in two cases. In the first case, all the connection cards that the users wield are concerned with the home operator. In the second case, connection cards that the users wield are concerned with the foreign operator. We ran the same scenario for both cases.

Figure 5.4 shows end-to-end latency for the first case together with standard deviation. End-to-end connection latency is the metric which is defined as the average time spent between sending the connection request by the client and end of the authentication process. For 600 nodes that are sending connection requests simultaneously, average latency is 1935 ms. As shown in Figure 5.4 with the linearly increasing number of connected clients, average latency and standard deviation is linearly increased, as well. Standard deviation of the average latency is high as 1660 ms.

The reason of this high standard deviation and linearly increasing connection latency is queue waiting time of the simultaneously sent connection request packets. However this is a stress test, and in real life this situation occurs rarely.

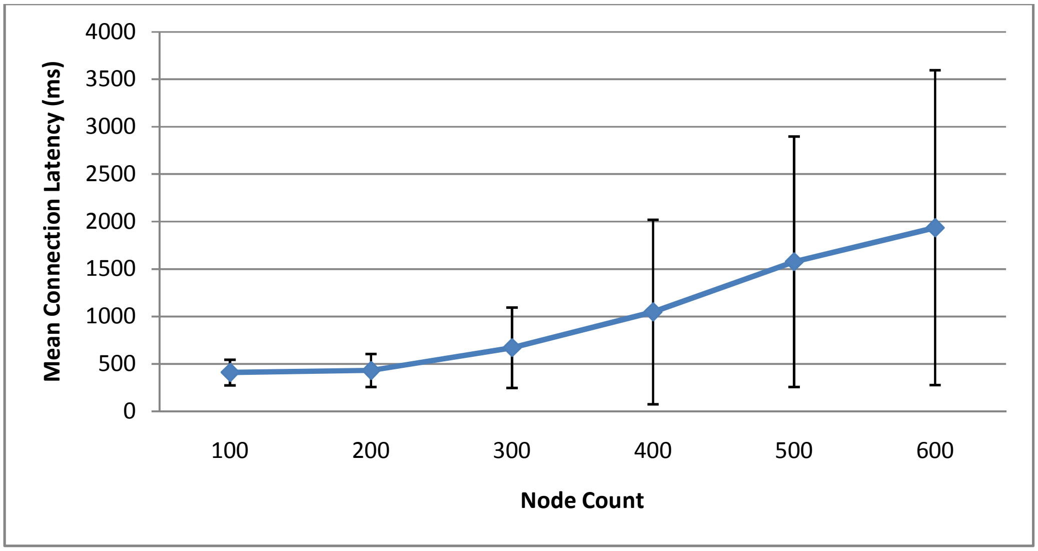

Figure 5.4. End-to-End Connection Establish Time in Burst Scenario (Home Network)

Mean server side service delay is shown in Figure 5.5. This metric is the average time spent between the arrivals of the connection request packet and processing of the packet. Hence, it includes the waiting time in the queue for processing. For 600 nodes it is approximately 15 ms. When we compared this with average end-to-end authentication latency, which is 1935 ms, we see that it is a tolerable delay. Although this delay is increasing linearly, as shown in Figure 5.5 increase rate of the delay is lower for the node count between 100 and 300, compared to node count between 300 and 600. When we check end-to-end connection latency for 300 nodes, it is 678 ms. This value is below 1 second, and it is a reasonable amount. This shows us that optimum system performance is provided for 300 clients.

When we simulate this scenario for foreign operator case, where all the clients wield connection cards of the foreign operator, results did not change much as shown in Figure 5.6. For 600 nodes average end-to-end authentication latency is 1945 ms. The reason of this is that as we have mentioned before, server side does not add a significant amount of processing time as compared to the APs. Hence, most of the delay is caused by the slowest network entity, which is AP. This shows us authentication latency can be decreased with network entities that have better processing units.

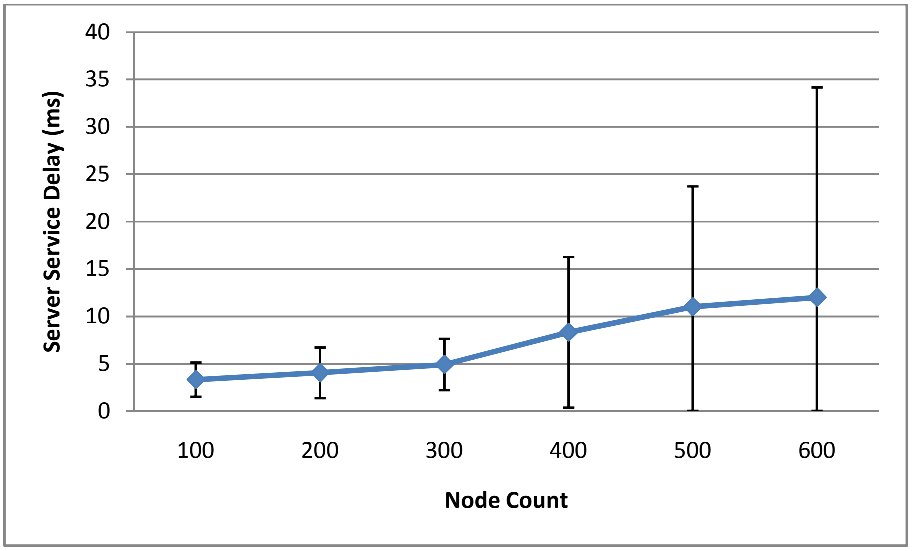

Figure 5.5. Server Service Delay

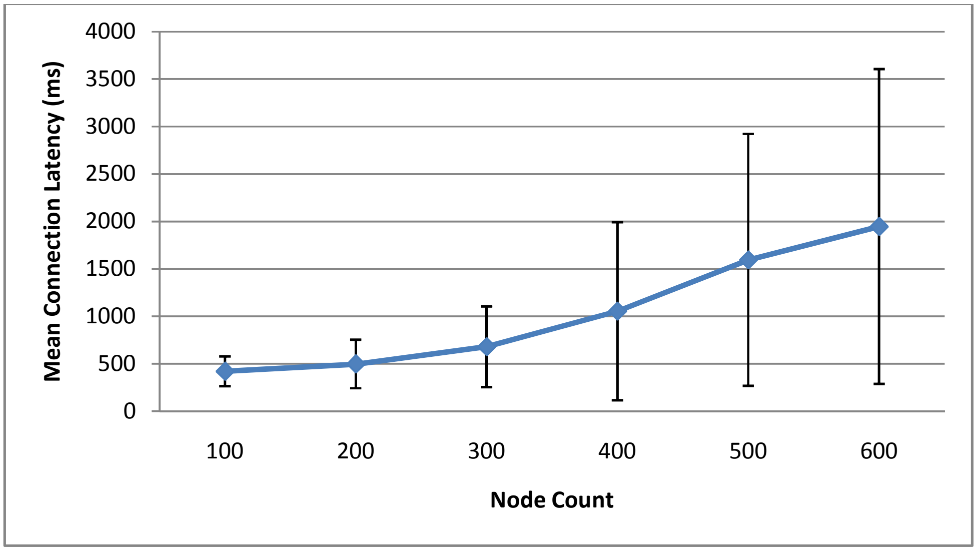

Figure 5.6. End-to-End Connection Establish Time in Rush Scenario (Foreign Network)

For calculation of the storage requirements of APs, in Figure 5.7 average and maximum connected client count per access point is shown. In our simulation for 600 clients, maximum number of client that an AP serves is 28. As expected, both average and maximum number of connected nodes increase linearly as the number of nodes increases.

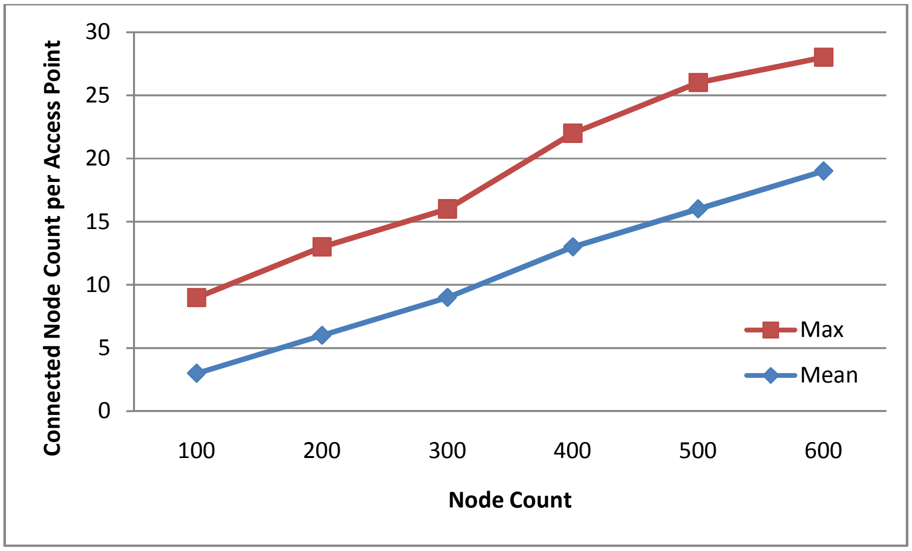

Figure 5.7. Average Connected Client Count per Access Point

5.2.2. Real-Life Scenario

In this scenario, the clients establish periodic TCP sessions with random intervals and durations throughout simulation. Simulations are run for 60 simulation minutes. We ran the same scenario for both home and foreign networks.

In each session, the amount of data that a user downloads is an exponentially distributed random variable with mean 20 MB. The first session of each client starts within the first 20 minutes, which is determined using uniform random distribution. The interval between two consecutive sessions of a user is also determined uniformly random within the range from 35 seconds to 15 minutes. If there is not any network activity for 30 seconds for a client, then it is automatically disconnected. In this way, we simulate reconnections. Moreover, if a node is disconnected, and there is a session request, then clients send connection requests for reconnection purposes.

As shown in Figure 5.8, if 40 nodes are running in the simulation, end-to-end connection latency is 532 ms in the home network, and standard deviation is 409 ms. In the situation, where an already sent connection packet is dropped, connection rerequest packets are sent. Connection latency includes these reconnection packets as well. This is the reason of high standard deviation.

In the rush scenario mentioned in previous section, end-to-end connection latency is 429 ms for 200 nodes. This value is smaller than the latency value that we have acquired here. This shows us the network traffic in the simulation is important. Hence, this situation leads us to run a third scenario, where clients simultaneously attempt to make connection requests in a loaded network. This scenario will be given in Section 5.2.3.

In Figure 5.9, maximum and mean numbers of simultaneously connected nodes are given. Not all the nodes are connected at the same time. Hence, we give mean number of connected nodes, and maximum reached number of connected nodes. As we mentioned before, we ran the same simulation scenario for both home and foreign operator. Hence, mean and maximum number of simultaneously connected nodes is same for both scenarios. In most of the cases; maximum number of simultaneously connected nodes reaches 50% of the total node count.

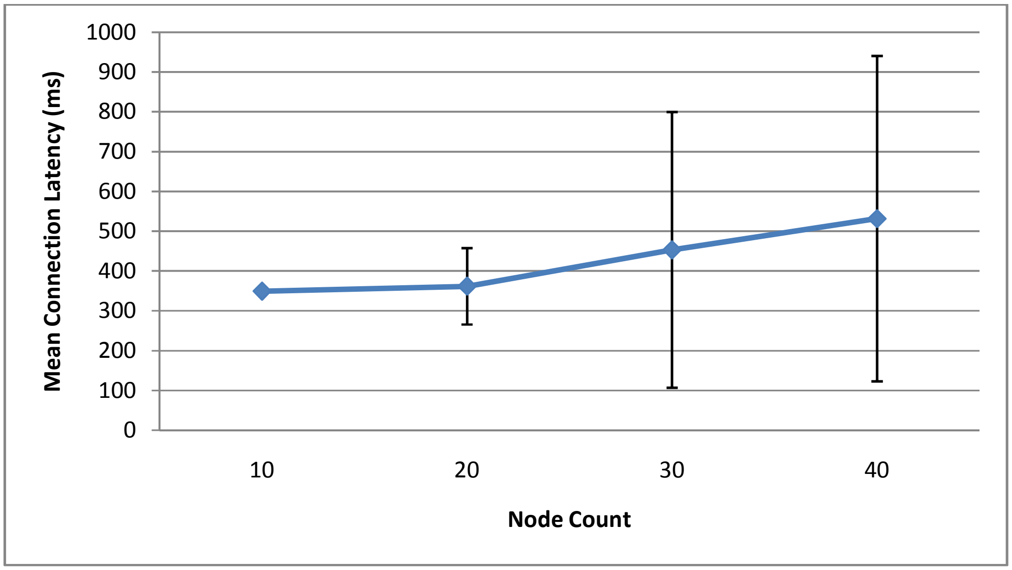

Figure 5.8. End-to-End Delay in Real-Life Scenario (Home Network)

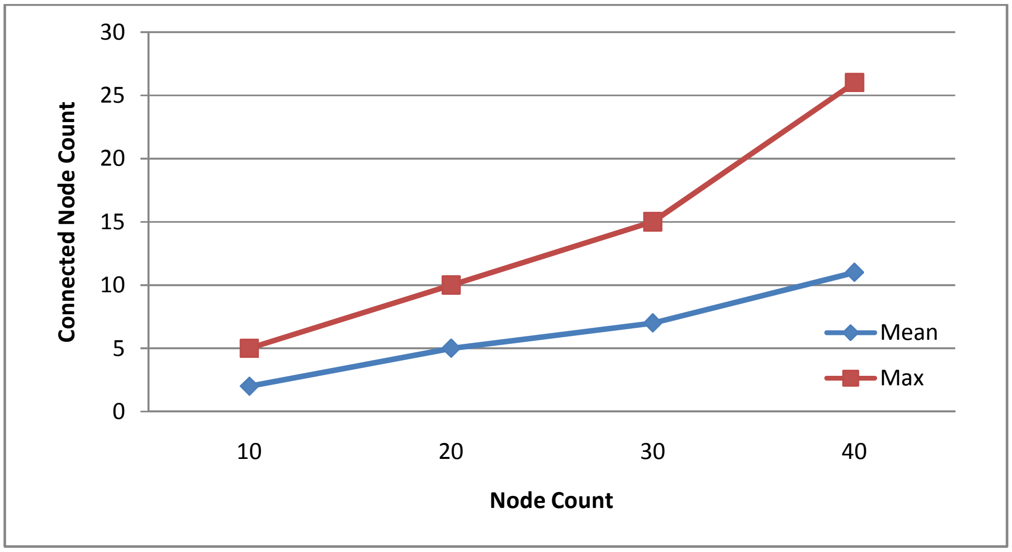

Figure 5.9. Connected Node Count Details (Home Network)

For foreign network, in Figure 5.10, when 40 nodes are running, average end-to-end connection latency is 551 ms. This value was 532 ms for home networks. Additional operation cost in the foreign operator server, and packet delivery time between home and foreign operator cause this difference.

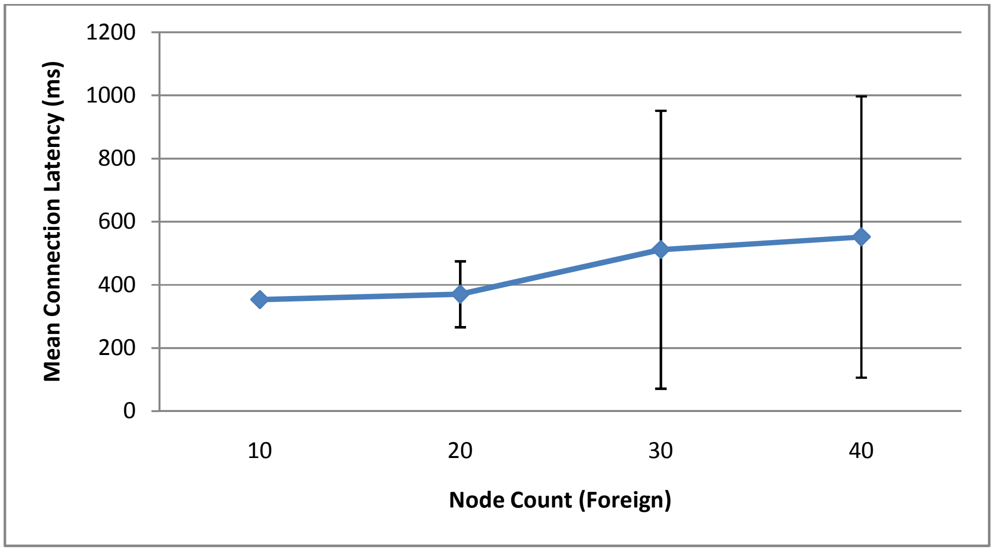

Figure 5.10. End-to-End Delay in Real-Life Scenario (Foreign Network)

5.2.3. Rush Scenario

In this scenario, we combine the two scenarios mentioned in the previous sections. As in real-life scenario, clients establish periodic TCP sessions with random intervals and durations throughout simulation. As in burst scenario, after 20 minutes, 200 nodes send connection requests simultaneously in 5 seconds. Hence, we measure how TCP packet transfers affect the connection latency in a stress test.

In this scenario simulations are run on the home network. Moreover, parameters are the same as the parameters at the real-life scenario. Connection latency results shown in Figure 5.11 only include the authentication delay in rush period within 5 seconds.

In burst scenario, average connection delay caused by 200 nodes is 429 ms as given in Figure 5.4. With the participation of active 40 nodes, this is increased to 648 ms. In Figure 5.11, real-life scenario results for the home operator, which was given in Figure 5.8, are given together with the rush scenario results. As shown in this figure, active clients change the connection latency with the same rate in both scenarios. In the simulations, we only measured the latency in the rush period within the 5 second range. Hence, TCP traffic is the only reason of the connection latency differences between these two scenarios.

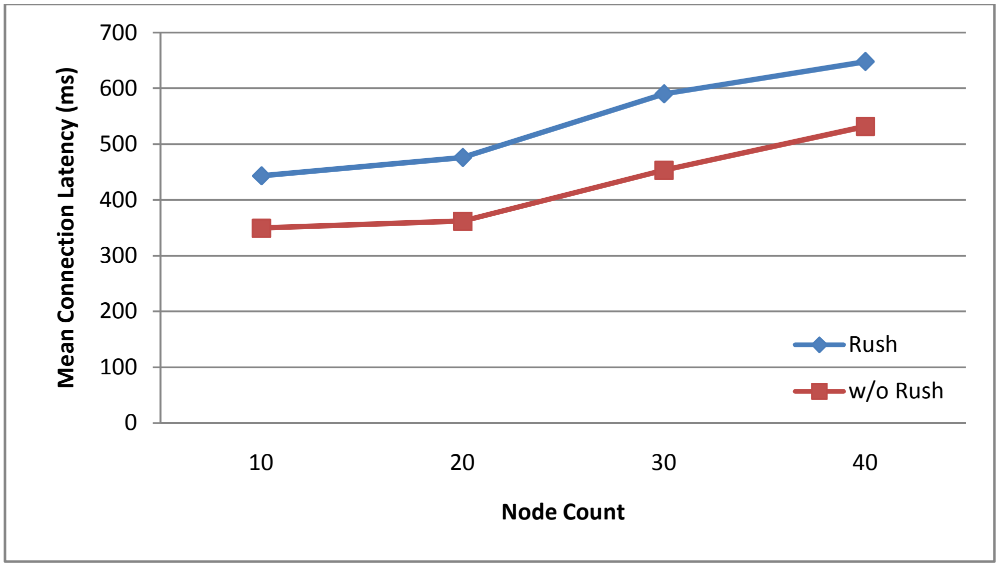

Figure 5.11. End-to-End Delay in Rush Scenario

5.2.4. Mobile Scenario

In this scenario, clients establish TCP sessions as in real-life scenario. There are five main differences between these two scenarios. These differences are listed below.

1. Each time a node disconnects from the system, it moves to a random point in the simulation area.
2. Instead of one operator, there are three operators that are giving service to the clients.
3. All the connection cards that are owned by different operators are distributed to the clients equally likely.

4. Each client connects to an AP with maximum transmission power irrespective of operator. That means roaming are possible.
5. Each AP belongs to one of the three operators. Ownership of the APs is equally likely among the operators.

We run a different simulation for this scenario. As in real-world scenario, we run four different simulations for 10, 20, 30 and 40 active clients. In Figure 5.12 average end-to-end connection latency and in Figure 5.13 average connected node count is given for mobile scenario. When we compare real-world (home network) and mobile scenarios, we see that average and maximum connected node count performance is the same for both scenarios when number of active clients is 10 and 20. However for larger active numbers the performance of mobile scenario is slightly worse. We also compare real-world scenario (home network) and mobile scenario for end-to-end latency metric. We see that the performance difference between these two scenarios is within +/- 10% range.

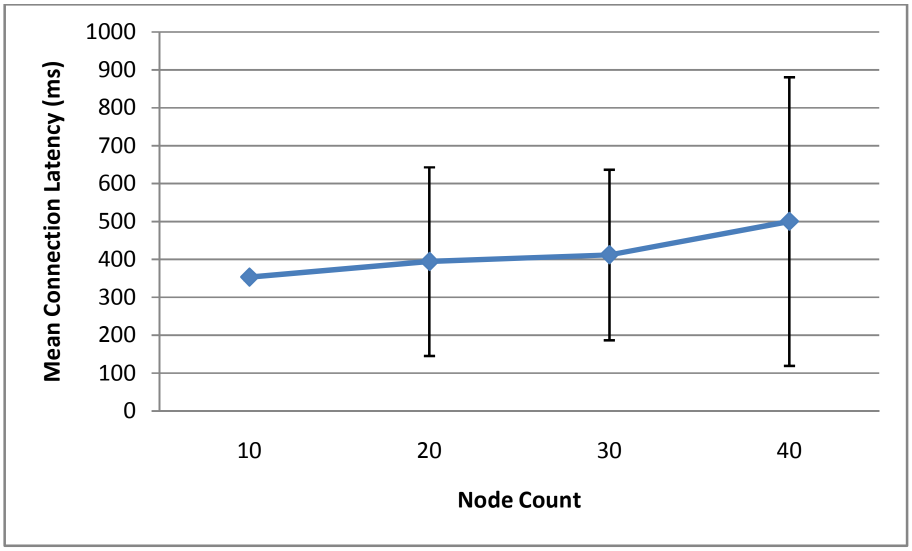

Figure 5.12. End-to-End Delay in Mobile Scenario

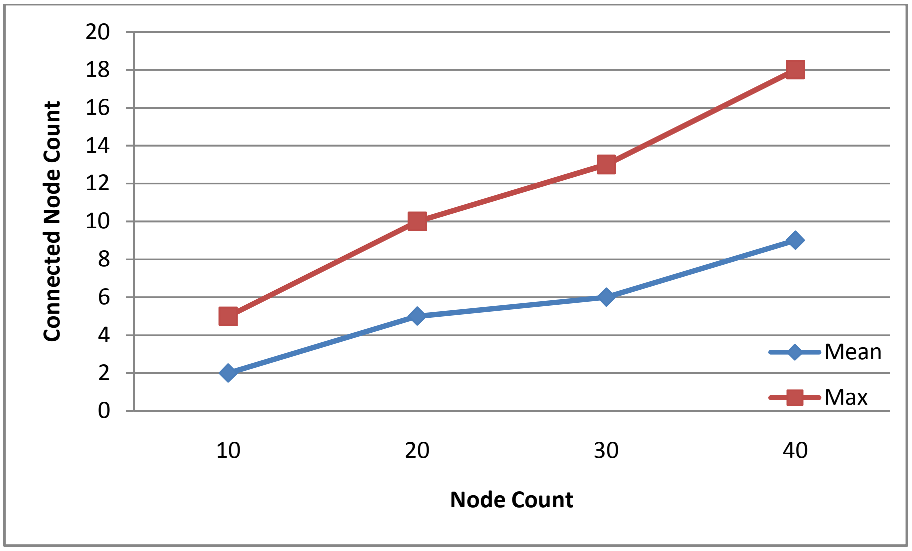

Figure 5.13. Connected Node Count Details in Mobile Scenario

6. CONCLUSIONS

In this thesis, we proposed a secure prepaid payment scheme for Wireless Mesh Networks. Our scheme is a micropayment one, which is based on tokens. Tokens are all contained in the connection cards. Tokens are generated as hash chains, by hashing an initial value several times.

We have simulated our protocol in four basic scenarios: first scenario is a stress test, which all the nodes sends connection request in a certain period of time. Second scenario is for testing a typical real-world case. Third scenario is a combination of these two scenarios, which clients make a rush in a real-world case. Fourth scenario is again a real-world case, but this time with three different operators and the clients are mobile.

We ran the first scenario for different number of nodes between 100 and 600. We consider two cases in this scenario. In the first case, all nodes connect to their home operator. In the second case all nodes connect to a foreign operator. In these two cases, for 300 nodes end-to-end authentication latency goes below one second. Most of the processing delay is caused by the *AP*. Hence, the results are similar in both cases.

For the real-world scenario, users start TCP sessions. TCP packets in the network cause longer end-to-end authentication latency. The mean latency for 40 nodes is 531 ms in the home operator, and 551 ms for the foreign operator. Some of the connection request packets are dropped, because of the congestion in the network. In this situation connection rerequest packets are sent by the clients. Our results include latency that is caused by the dropped connection request packets.

In the rush scenario, for 200 newly added nodes and 40 existing active nodes, we have seen average end-to-end authentication latency is increased from 429 ms to 648 ms. Thus, we conclude that for this case that we consider, existing network traffic increase the authentication latency 50%.

Our simulations for the mobile scenario showed that mobility does not cost too much overhead for the performance metrics.

The results show that, roaming processing times added to the end-to-end connection latency is tolerable. Active client count is the main criterion that affects latency.

7. REFERENCES

[1] Akyildiz, I. F., Wang, X., and Wang, W. (2005) Wireless mesh networks: a survey, *Computer Networks and ISDN Systems*, 47(4): 445-487.

[2] Zhang, Y., and Fang, Y. (2006) A secure authentication and billing architecture for Wireless Mesh Networks, *Wireless Networks*, 13(5): 663-678, Springer Netherlands.

[3] Lamport, L. (1981) Password authentication with insecure communication, *Proceedings of Commun. ACM,* vol. 24, no. 11, pp. 770-772.

[4] Rivest, R., Shamir, A., and Adleman, L. (1978) A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, 21(2): 120–126.

[5] Diffie, W., and Hellman, M. E. (1976) New directions in cryptography, *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644-654.

[6] Trappe, W., and Washington, L. (2006) Introduction to cryptography with coding theory, *Person Education, Inc.*

[7] Stallings, W. (2006) Cryptography and network security, *Person Education, Inc.*

[8] Salem, N. B., and Hubaux J.P. (2006) Securing wireless mesh networks, *IEEE Wireless Communications Magazine,* vol. 13, no. 2, pp. 50-55.

[9] Soliman, H., Castelluccia C., and Malki K. E., Bellier. L. (2005) Hierarchical mobile IPv6 mobility management (HMIPv6), RFC 4140.

[10] Efstathiou, E., Frangoudis, P., and Polyzos, G. (2006) Stimulating Participation in Wireless Community Networks, *IEEE INFOCOM, 2006,* Barcelona, Spain.

[11] OMNET++ Discrete Event Simulator, www.omnetpp.org

[12] INET Framework, http://inet.omnetpp.org/

[13] MiXiM: Mixed Simulator, http://mixim.sourceforge.net/

[14] Oversim: The Overlay Simulation Framework, http://www.oversim.org/

[15] Crypto++ Library 5.6.0, http://www.cryptopp.com/

[16] Deng, L., and Kuzmanovic, A., (2009) A feeder-carrier-based internet user accountability service, *Northwestern University Technical Report,* http://networks.cs.northwestern.edu/susinet/TR-09-12.pdf

[17] Zhang, Y., Zheng, J., and Hu, H. (2009) Security in wireless mesh networks, *Auerbach Publications*.

[18] Krawczyk, H., Bellare, M., and Canetti R. (1997) HMAC: Keyed-Hashing for Message Authentication, RFC 2104.

[19] Deering, S., and Hinden R. (1998) Internet Protocol Version 6 Specification, RFC 2460.

[20] Vaughan-Nichols S.J., (2004) Achieving wireless broadband with WiMax, *IEEE Computer*, vol. 37, no.6., pp. 10-13.

[21] Robshaw, M. J. B. (1995) Stream ciphers, *RSA Laboratories Technical Report.*

[22] Schneier, B. (1993) Description of a new variable-length key, 64-bit block cipher (Blowfish), *Fast Software Encryption, Cambridge Security Workshop*, LNCS 809, pp. 191-204.

[23] FIPS PUB 46-3 (1999) Data Encryption Standard (DES), http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[24] FIPS PUB 197 (2001) Announcing the Advanced Encryption Standard (AES), http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[25] FIPS PUB 186-3 (1994) Digital Signature Standard (DSS) - CSRC, http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf

[26] Koblitz, N. (1987) Elliptic Curve Cryptosystems, *Mathematics of Computation* vol.48, no. 177, pp. 203-209.

[27] Yakovyna, V., Fedasyuk, D., Seniv M., Bilas O. (2007) The performance testing of RSA algorithm software realization, *CAD Systems in Microelectronics*, *CADSM '07*, pp. 390-392, Polyana, UKRAINE