SECURE AND SEAMLESS PREPAYMENT FOR WIRELESS MESH NETWORKS

by

SERHAT CAN LELOĞLU

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

Sabanci University
January 2013

SECURE AND SEAMLESS PREPAYMENT 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	....

SECURE AND SEAMLESS PRE-PAYMENT FOR WIRELESS MESH NETWORKS

Serhat Can Leloğlu
Computer Science and Engineering, MS Thesis, 2013
Thesis Supervisor: Assoc. Prof. Albert Levi

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

Abstract

Wireless Mesh Network (WMN) is multi-hop high-speed networking technology for broadband access. Compared to conventional network service providing systems, WMNs are easy to deploy and cost-effective. In this thesis, we propose a secure and seamless pre-payment system for the Internet access through WMNs (SSPayWMN).

Practical payment systems for network access generally depend on trustworthiness of service provider. However, in real life, service providers may unintentionally overcharge their clients. This misbehavior in the system may cause disputes between the clients and the service providers. Even if the service provider is rightful, it is very difficult to convince the customer since the service providers generally do not have justifiable proofs that can easily be denied by the clients.

The main goal of SSPayWMN is to provide a secure payment scheme, which is fair to both operators and clients. Using cryptographic tools and techniques, all system entities are able to authenticate each other and provide/get service in an undeniable way. Moreover, SSPayWMN provides privacy and untraceability in order not to track down particular user's network activities.

We implemented SSPayWMN on a network simulator (ns-3) and performed performance evaluation to understand the latency caused by the system's protocols. Our results show that our protocols achieve low steady state latency and in overall put very little burden on the system.

KABLOSUZ ÖRGÜ AĞLARI İÇİN GÜVENLİ VE KESİNTİSİZ ÖN ÖDEMELİ SİSTEM

Serhat Can Leloğlu

Bilgisayar Bilimi ve Mühendisliği, Yüksek Lisans Tezi, 2013

Tez Danışmanı: Doç. Dr. Albert Levi

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

Özet

Kablosuz Örgü Ağları (KÖA) çok sekmeli, yüksek bağlantı hızı sağlayabilen ağ teknolojileridir. Alışılagelmiş ağ servisi sağlayan sistemlere nazaran kurulumu kolay ve ekonomiktir. Bu tezde, güvenli ve kesintisiz bir ön ödeme sistemini (SSPayWMN) anlatıyoruz.

Ağ bağlantısı için tasarlanmış ödeme sistemleri genelde servis sağlayıcısının güvenilirliğinden kuşku duymazlar. Fakat servis sağlayıcılar bazen istemeden de olsa fazladan ücretlendirme yapabilirler. Bu tarz durumlar müşteri ve servis sağlayıcı arasında anlaşmazlığa sebebiyet verir. Servis sağlayıcılar haklı dahi olsalar bunu müşteriye kanıtlayacak inkar edilemez kanıtları olmadığı için müşteriyi ikna edemezler.

SSPayWMN'in asıl amacı güvenli bir ödeme sistemi kurmanın yanı sıra hem servis sağlayıcı hem de müşteri için adil bir ödeme sistemi sağlamaktır. Kriptografik algoritmaları ve teknikleri kullanarak, sistemin bütün elemanları kimliklerini kanıtlayabilir ve sonradan inkar edilemeyecek şekilde servis sağlayabilir veya servislerden faydalanabilirler. Bunun yanında, SSPayWMN kullanıcıların mahremiyetini ve sistemdeki hareketlerinin takip edilememesini sağlar.

SSPayWMN bir ağ simülatörü (ns-3) üzerinde test edildi ve performansı değerlendirildi. Sistem tarafından sebep olunmuş gecikmeler hesaplandı. Sonuçlar protokollerimizin düşük gecikme değerlerinde dengeyi yakaladıklarını göstermiştir ve protokollerin sisteme minimum oranda yük getirdiğini kanıtlamıştır.

To my dear family

ACKNOWLEDGEMENTS

I would like to thank my thesis supervisor, Albert Levi, who has offered me this project at the beginning of my master studies. He has always been very helpful and understanding.

I would like to thank Yücel Saygın for introducing a new area of computer science to me with Information Retrieval and Data Mining courses. He was very kind by accepting to join the jury.

I would like to thank Erkay Savaş for giving me my cryptographic basis in my undergraduate studies. I would like to thank him for being a part of my jury.

I also thank Özgür Erçetin and Cemal Yılmaz for devoting their time for jury during their tight schedule.

I would like to thank my wife Suna Yeltekin Leloğlu, my mother Zerrin Leloğlu and my sister Seden Leloğlu Seçilir, for their support on every step of my life.

I would like to thank my father Osman Nuri Leloğlu for everything he did for me. He will always live in memory.

I would like to thank to my niece Birce Seçilir, for her calming effect. I wish she would achieve much more success than I did, when she comes to my age.

I would like to thank Can Yücel for explaining the project to me.

Finally, I thank Türk Telekom for funding this project under which I got my scholarship.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

WMNs [1] offer broadband network access with high-speed network connection. WMNs are easy to deploy and cost effective compared to conventional Internet service providing infrastructures such as high-powered servers. Mesh networks dynamically organize themselves and do not need a centralized element; they can be considered as a subset of ad hoc networks. Mesh nodes deliver packets from source to destination in a multi-hop manner, thus they extend the network coverage. WMNs could interconnect with conventional Wi-Fi networks and also other network types such as WiMax [2], ZigBee [3] and 3G-radio access [4].

WMNs are cost effective and they extend the network coverage, thus they could be used for network service providing. A necessity for a billing system automatically emerges for network service providing. Such systems require secure billing protocols that ensure anonymity and confidentiality of the client. A secure billing scheme should guarantee that a client could not get service without payment and service provider cannot overcharge the client.

There has been extensive research about micropayment systems for network service providing systems. Rivest and Shamir [5] proposed Pay-Word for micropayment purposes. Their work opened a new research area and many modifications and improvements [6] were performed on their scheme. New algorithms increased performance and flexibility. The research on micropayment schemes are still ongoing, there are some modern approaches on micropayment such as [7]. In [7] authors suggested to use hash chains [7] and public key cryptography to provide confidentiality, anonymity and untraceability for clients.

Micro-payment systems generally put partial or full trust in network service providers. This application gives service providers power to force clients to stick with them, which decreases the flexibility of the system. Furthermore, network service providers could overcharge the clients; there is a necessity for some system to protect client’s rights. Clients should be able to object an overcharge situation with undeniable proofs. A system with these properties should also ensure good performance with confidentiality, anonymity and untraceability for the clients.

SSPayWMN implements a prepaid billing scheme with simple structure and trust models. Authentication, confidentiality, non-repudiation and client privacy are taken into the consideration in the system design. SSPayWMN employs some cryptographic primitives to ensure system security. The billing system depends on hash chains and uses every element of the hash chain as a token, which buys time intervals for Internet service. SSPayWMN employs a Trusted Third Party (TTP), which forces honest usage of the system by every party.

1.1. CONTRIBUTION OF THE THESIS

In this thesis we propose a secure and seamless prepayment system for WMNs. The system depends on an e-currency that is sold by a trusted third party. E-currency in SSPayWMN is generated in a unique way. SSPayWMN covers anonymity and untraceability lacks of similar pre-payment systems.

Authentication, confidentiality, non-repudiation, fraud protection is provided in the system. The users are not being able to deny using credits for the services that are received; the operator is not being able to charge more than the usage amount. Additionally, inter-operator settlement is performed in a secure way such that each operator has cryptographic proofs provided service for every client in the system. In order to provide privacy of clients, our scheme provides untraceability such that no unauthorized entity will be able to track down a particular user with certain limits.

We evaluated the performance of SSPayWMN using network simulations. These simulations are conducted in two groups. The first group is unit tests, aims to simulate protocols of protocols of SSPayWMN, independent from each other to analyze the delay caused by protocols. Unit test results show that all of SSPayWMN protocols reach steady state performance. SSPayWMN is designed to reckon with real-life challenges such as stable Internet service during client mobility and overweight network service demand in rush hours. In the second simulation group, called real-life scenario simulations, the clients are selected considering human behavior and they are categorized. In real-life scenario simulations the system also reached steady state, which ensures system stability.

1.2. ORGANIZATION OF THE THESIS

The organization of this thesis is as follows. A brief introduction will be given in Section 1. We will give background information on wireless mesh networks and cryptographic primitives in Section 2; moreover, we will give literature survey on billing systems and related works of SSPayWMN in the same section. In Section 3, requirements and protocols of the system are described. In Section 4, we will present the simulation environment and simulation results. We will discuss the success of SSPayWMN in Section 5. Finally, we will give the conclusions in Section 6.

2 BACKGROUND INFORMATION AND LITERATURE SURVEY

In this section we give background information about WMNs in Section 2.1. Section 2.2 gives brief background information on cryptographic primitives. A literature survey will be given and related works of SSPayWMN will be explained in Section 2.3. The proposed system is the improved version of a previous project [8]; similarities and differences with [8] will be explained in Section 2.4.

2.1 BACKGROUND ON WIRELESS MESH NETWORKS

Wireless Mesh Network (WMN) is a multi-hop type of wireless networking, designed as an alternative to traditional centralized wireless networking achieved by mesh routers. Mesh routers and mesh network clients form up WMNs. Each mesh node functions as a host and also as a router, relaying packets on behalf of other nodes, connecting nodes that are not located within the transmission range of each other. WMNs create ad hoc networks, which are dynamically self-organized and self configured. WMNs are easy to deploy and cost-effective systems, they are easy to maintain and provide robustness and reliable service coverage.

WMNs comprise of two types of nodes: mesh routers and mesh clients. A wireless mesh router provides mesh networking by using routing functions that do not exist in common wireless routers with gateway/repeater capabilities. Mesh routers have multiple wireless interfaces to expand flexibility of WMNs. Mesh routers in WMNs achieve wider coverage compared with conventional routers by using multi-hop technology with lower transmission power. Moreover it is possible to hypothesize improved scalability by optimizing the medium access control (MAC) protocol in a mesh router.

2.1.1. Network Architecture

WMNs are categorized into three considering the operation of the nodes in the network. These categories are infrastructure, client and hybrid WMNs.

Infrastructure/Backbone WMNs: The architecture of this WMN is shown in Figure 2.1. Both wireless and wired networks comprise infrastructure WMNs. Dash lines depict wireless connections; whereas, solid lines depict wired communications. Mesh routers establish an infrastructure to mesh clients to connect. The infrastructure is a cloud from the clients' point of view. It is a black box that delivers packets originated from the clients to the gateways.

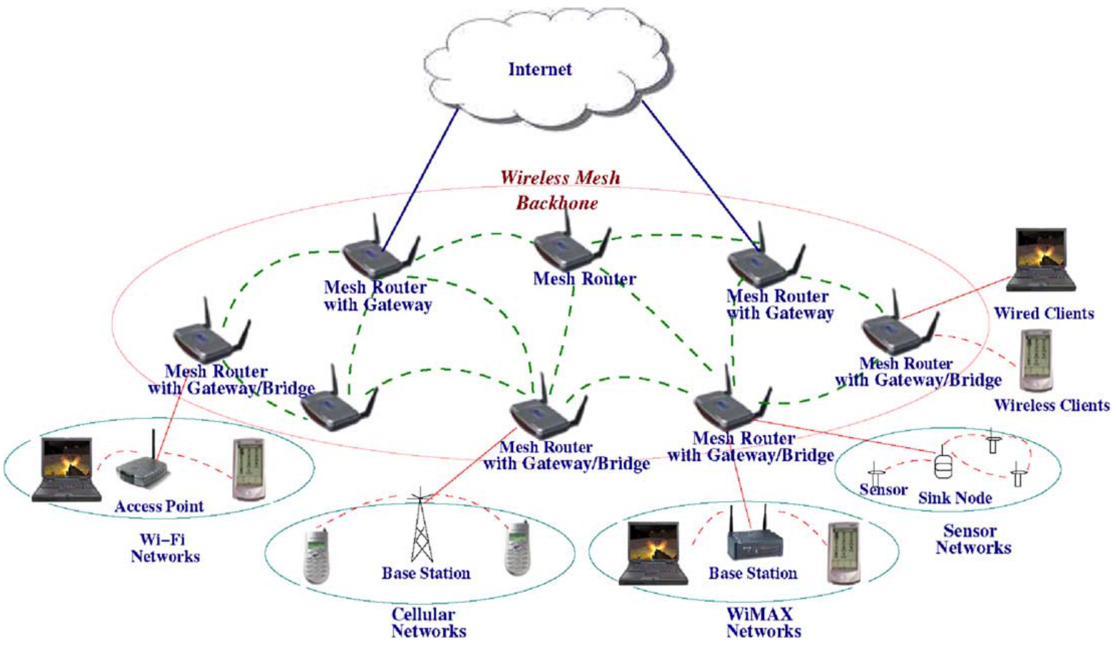

Figure 2.1. Infrastructure/Backbone WMNs. (taken from [1])

The mesh routers are self-configured and self-healing. In a case of node addition or removal, mesh backbone configures itself by forming up neighborhood. Additionally, mesh routers could connect to the Internet with gateway functionality. Infrastructure meshing provides easy to access to Internet by forming up clouds for clients. Bridging and inter-networking functionalities of WMNs enable clients to connect to mesh backbone with conventional Wi-Fi or cellular devices and also via Ethernet links. As depicted in Figure 2.1, base stations could also connect to mesh backbones, which provide Internet connectivity for all the clients of base stations.

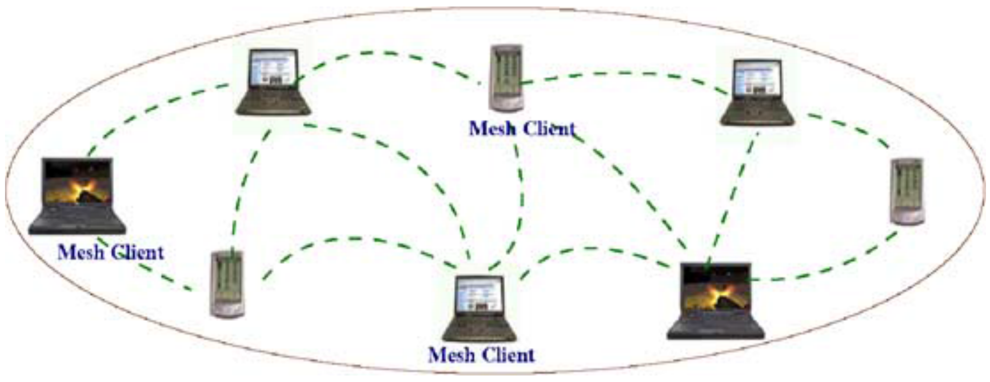

Figure 2.2. Client WMNs (taken from [1])

Client WMNs: Client meshing is a subset of Infrastructure meshing. As previously explained mesh routers establish a backbone for mesh clients, however in client meshing case the whole network is a backbone and whoever wants to join the network has to be a part of the backbone and provide routing functionality. As shown in Figure 2.2, client meshing is a commune type of networking.

Hybrid WMNs: This architecture is combination of two previously explained mesh architectures. Mesh clients can access Internet through mesh backbone moreover they can communicate with each other by using a simple ad hoc network.

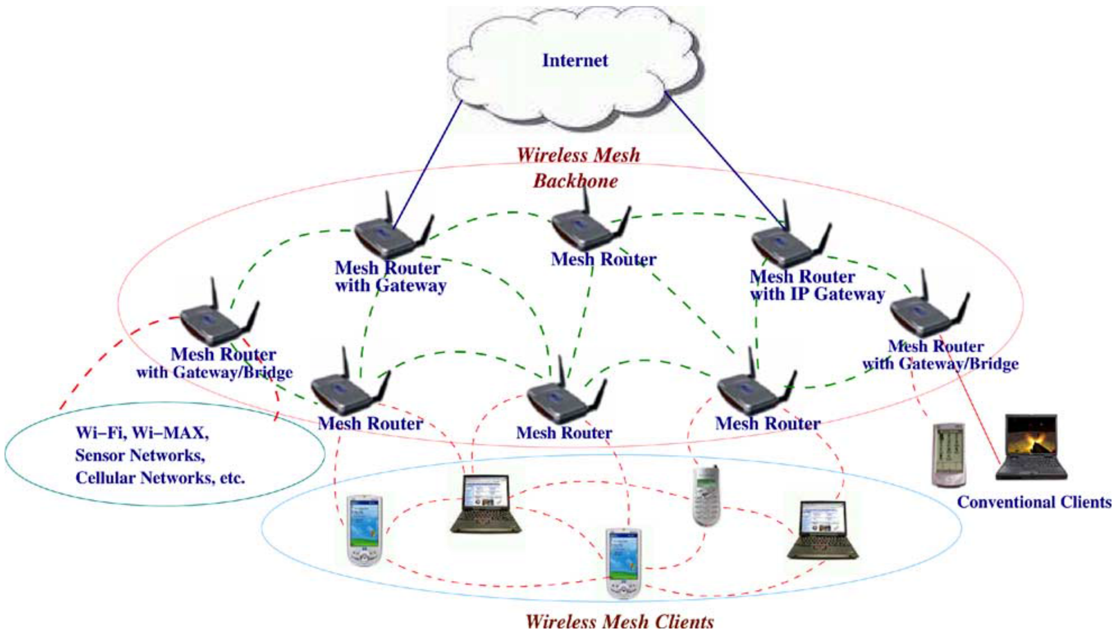
Internet
Wireless Mesh Backbone
Mesh Router with Gateway
Mesh Router
Mesh Router with IP Gateway
Mesh Router with Gateway/Bridge
Mesh Router with Gateway/Bridge
Mesh Router
Mesh Router
Mesh Router
Wi–Fi, Wi–MAX, Sensor Networks, Cellular Networks, etc.
Conventional Clients
Wireless Mesh Clients

Figure 2.3. Hybrid WMNs (taken from [1])

As shown in Figure 2.3, mesh backbone provides Internet connectivity; whereas, Client WMNs provide connectivity to mesh backbone for distant mesh clients.

2.1.2. Characteristics of Wireless Mesh Networks

Characteristics of WMNs are explained as follows:

- *Multi-hop Wireless Network:* Main accomplishment of WMNs is providing extended wireless network coverage without increasing transmission power or additional antennas.
- *Support for Ad hoc Networking:* WMNs provide flexible networking, which has abilities like self-configuring and self healing. Deployment, node addition and removal are easy to accomplish since mesh routers form up routing paths by themselves.
- *Mobile Dependence on the Type of Mesh Nodes:* Mesh routers usually do not change their locations, whereas mesh clients are assumed to be mobile.
- *Multiple Types of Network Access:* Mesh routers are accessible via IEEE 802.11 protocols and also from peer-to-peer protocols.
- *Dependence of Power-Consumption Constraints on the Type of Mesh Nodes:* Mesh routers usually do not have power-consumption constraints but it is advisable for mesh clients to have some forms of power consumption techniques.
- *Compatibility and Interoperability with Existing Wireless Networks:* WMNs are compatible with Wi-Fi networks as well as other types of networks.

2.2. BACKGROUND ON CRYPTOGRAPHIC ALGORITHMS

To establish a secure system, cryptographic algorithms are employed. A brief explanation and introduction to cryptographic primitives are provided in this section.

In following sections, hash functions, hash chains and HMAC functions are explained. Moreover symmetric cryptography is described. Finally public key cryptography is explained at the end of this section.

2.2.1. Hash Functions

Hash functions [9] are irreversible mathematical functions that map input strings of variable length to fixed sized output strings. Hash functions operate fast and it is infeasible to find the input string of a given hash value.

A cryptographic hash function should have at least the following three properties:

- Pre-image resistance: Given a hash value v, it should be infeasible to find the input string x such as $v = h\ (x)$ (hash of x).
- Second pre-image resistance: Given an input x_1, it should be infeasible to find another input string x_2 such that $h(\ x_1) = h\ (x_2)$.
- Collision resistance: It should be infeasible to find two input strings x_1 and x_2 such that $h(x_1) = \text{h}(x_2)$.

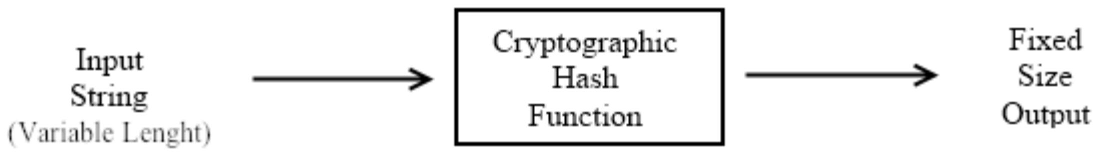

Figure 2.4. Hash Function Example

Figure 2.4 depicts the hash function flow. The function maps a message to a fixed size string. The output length depends on the hash algorithm; various hash algorithms have different output sizes.

Once well regarded and now outdated hash algorithms are MD5 [10] and SHA1 [11]. In parallel with improvements of cryptanalytic attacks and computational power, these hash functions are not considered as secure anymore. Up to date, popular and well-regarded hash functions are SHA2 [12], which is a set of SHA-256, SHA-384 and SHA-512 and SHA-3 [13] also known as Keccak.

2.2.2. Hash Chains

Applying a hash algorithm to a $seed$ value and using the output as an input for the next hash function form a hash chain. Every output of a hash algorithm represents a link in the chain. Length of the hash chain [14] is determined by the number of times the hash algorithm is executed.

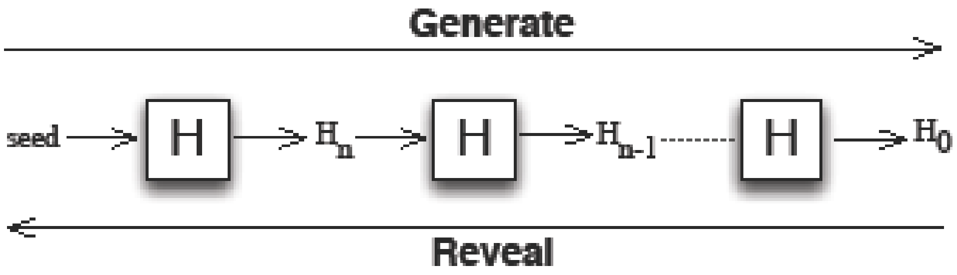

Figure 2.5. Hash Chain Depiction and Usage

Since hash functions are irreversible, as shown in Figure 2.5, it is easy to go forward in the chain but it is not computationally feasible to go backwards.

A hash chain with n elements is denoted as:

$$H_n = h(seed);\ H_{n-1} = h(H_n); \dots;\ H_0 = h(H_1)$$

Knowing the first link in the chain, which is H_0, gives the opportunity to verify the following links in the chain as well. If one could establish a system, successful at distributing the first link in the chain, it is feasible to use hash chains as future keys for other cryptographic functions or tokens for micropayment.

Hash chains are easy to deploy and cost effective; therefore, they are widely used in prepayment systems, especially for systems that have delicacy for computational delay.

2.2.3. HMAC Functions

A Message Authentication Code (MAC) [9][15] is a short message, which is calculated to provide authentication and message integrity. A MAC algorithm is a function, which generates a MAC using a secret key. Calculating the MAC of the message and comparing the received MAC check the integrity of the message. If the values are equal then the integrity of the message is verified. Authentication is also provided because a party, who is able to calculate a specific MAC value, proves that she knows the secret key.

HMAC [16] is one of the most popular MAC functions that use a keyed hash function. Hash functions are fairly fast algorithms, which makes HMAC a fast algorithm.

2.2.4. Symmetric Cryptography

Symmetric cryptography is the oldest kind of cryptographic primitive, which provides confidentiality. This primitive employs shared secret keys between two parties. These secret keys are used in encryption of plaintexts and decryption of ciphertexts. The security level of a symmetric cryptographic algorithm mostly depends on the key size. Modern algorithms use at least 128-bit long keys.

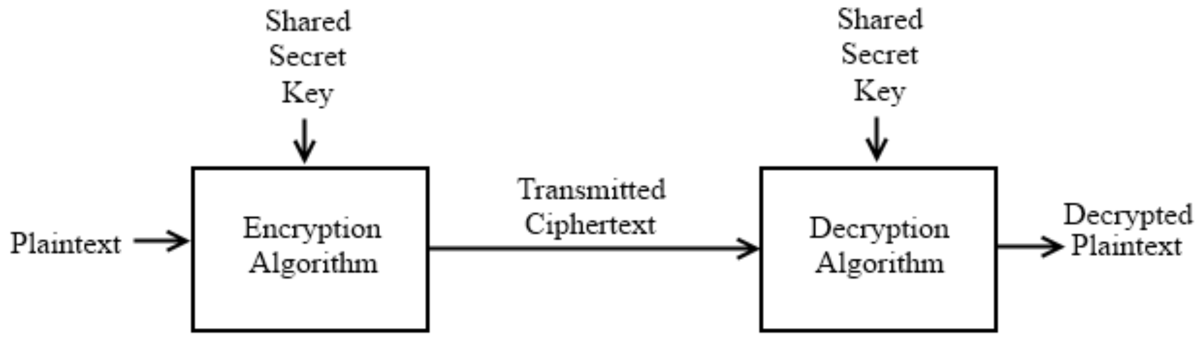

Figure 2.6. Symmetric Key Cryptography

As shown in Figure 2.6, a plaintext is used as a parameter with the shared secret key in an encryption algorithm. Encrypted data is transmitted through an insecure medium. The receiver of the encrypted message decrypts the cipher text by using the shared secret key in the decryption algorithm.

Modern symmetric cryptographic functions could be categorized under two classes, which are *stream ciphers* and *block ciphers*. *Stream ciphers* encrypt data byte by byte. RC4 [17] is an example for *stream ciphers*. Secure Socket Layer (SSL) and Wired Equivalent Privacy (WEP) employ RC4. On the other hand, *block ciphers* encrypt an input data as fixed sized blocks, and produces fixed sized outputs. Data Encryption Standard (DES) [18] is once considered as the most popular symmetric key cryptography standard but it is not considered as secure any longer. RC5 [19] and Blowfish [20] are some other examples of symmetric key cryptography. Advanced Encryption Standard (AES) [12] is an example of modern symmetric cryptography algorithms.

2.2.5. Public Key Cryptography

Public Key Cryptosystem (PKC) differs from Symmetric Key Cryptosystem according to number of keys used. PKC uses two separate keys; one of them is the public key the second one is the private key. The owner secretly keeps the private key; while broadcasting the public key. It is computationally infeasible to calculate private key by exploiting the public key.

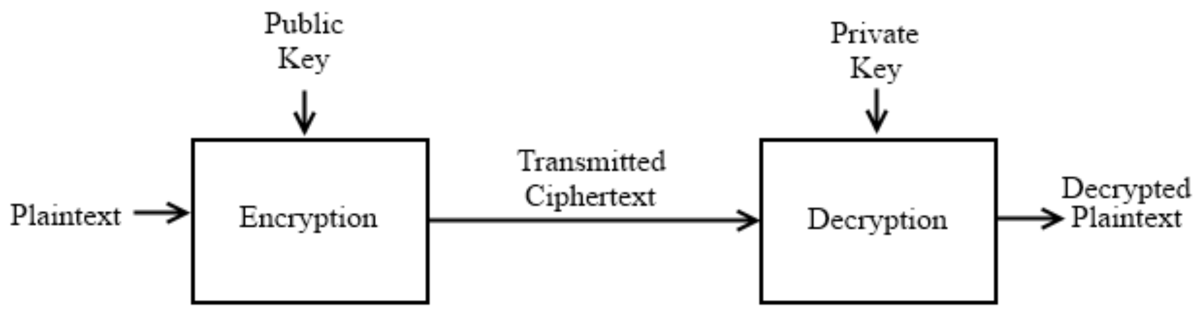

Figure 2.7. Public Key Encryption

PKC is used for confidentiality purposes, i.e. for encryption and decryption. It is also used for digital signing and verification. The type of encryption key defines the purpose of the algorithm. If the sender uses the public key for encryption then the algorithm operates for confidentiality purposes as shown in Figure 2.7.

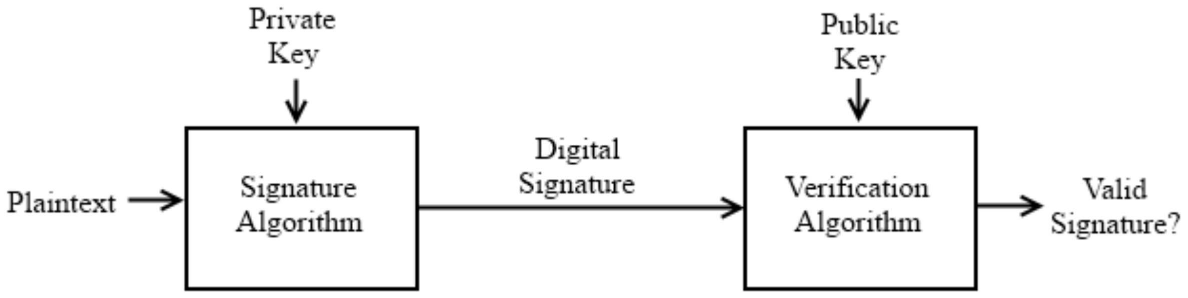

Figure 2.8. Validating a Signature

Authentication and non-repudiation are provided by using private key as the encryption key as depicted in Figure 2.8. Since no one could know the private key of the owner, only private key owner could produce the digital signature with this private key. These signatures could be verified using the public key. Since the public keys are broadcast, anyone could verify the digitally signed plaintext.

Some of the widely known asymmetric key cryptographic algorithms are Elliptic Curve Cryptography (ECC) [22][23], Digital Signature Algorithm (DSA) [24], ElGamal [25] and the most known one is RSA [26] algorithm.

2.3. LITERATURE OF PAYMENT SYSTEMS AND SIMILAR WORKS OF SSPAYWMN

There has been extensive research on billing systems for network access. In this section a brief explanation will be given about similar works of SSPayWMN.

E-Payment schemes [27] could be grouped under two classes:

- Online Payment: The seller checks the payment by consulting a trusted party before verifying it.
- Offline Payment: The seller does not check the payment before serving the client.

Online payment schemes could be subdivided into two subgroups:

a Payment by Transaction: Buyer and seller do not need a previous arrangement before the payment.

b Payment by Account: There exists an account or a previous engagement between seller and buyer.

Micro-payment systems are built on e-payment systems. Micro-payment schemes generally employ a type of electronic currency (e-cash/e-coin). General requirements [28] for micro-payment schemes are:

- Anonymity: A payment should not reveal buyer's identity.
- Divisibility: The amount of money should be divided into e-currency. The e-currency should have a unit value.
- Transference: The e-currency should have approximately equal value when it is transferred between parties.
- Prevention of Double Spending: Clients should not be able to use e-currency for a second time.

Rivest and Shamir [5] introduced a simple micro-payment scheme called Pay-Word. Their protocol uses RSA algorithm for public key cryptography and hash-chains as payment tokens. In Pay-Word protocol the buyer generates a hash chain for a specific seller before making a purchase. The seller checks if the user is an authenticated Pay-Word user and the hash chain is indeed generated by this particular user. The amount of purchase is determined by the hash operation count performed on a value. The buyer selects the value and after the purchase deletes the hash chain. The seller collects the deserved money from the Bank, which is a trusted third party in the system.

Pay-word is a credit-based payment scheme. In [6], the drawbacks of the system are pointed. [6] states that, the main drawback is seller specific hash chains. Clients should store information about the sellers to be able to generate seller specific hash chains. Every time a new seller is added to a system the clients should be notified. In the existence of large amount of sellers the system becomes infeasible. Furthermore, the user should store different hash chains for every seller and store the last used hash token for all of the hash chains.

Rivest and Shamir opened up an exciting field of research since then there has been some modifications on their work. In [6], authors propose a new algorithm, which increases the performance of the previous algorithm. They suggest an efficient payment system that uses RSA-typed blind signature [29]. The system supports anonymity and untraceable payments. The system supports one-way authentication; client is authenticated to the network but the network does not authenticate itself to the client. The system is useful for payment systems dealing with low amount of payments count.

Hwang and Sung [30] proposed a new micropayment scheme in 2006. Their micro-payment scheme uses one-way property of hash chains and elliptic curve cryptography for blind signatures and for the purpose of providing anonymity. The system has three phases:

1. Registration Phase: Buyer and seller registers to the system by authenticating themselves to the trusted party.
2. Blinding Phase: The buyer executes a withdrawal protocol with the trusted party before purchasing for any service.
3. Transaction Phase: The buyer requests service from the seller and send the e-currency in this stage.
4. Redemption Phase: The seller communicates with the trusted party to redeem the deserved money.

The proposed scheme of Hwang and Sung remains secure and it provides anonymity as well. However, as [6] states, the usage of elliptic curve cryptography in e-payment systems slows down the entire system.

Up to now, general e-payment schemes have been described. In this thesis, we focused on micropayment schemes for broadband Internet access using WMNs. In the rest of this section, we explain payment schemes for broadband network access in WMNs and other network types.

Chen, Jan, and Chen [31] proposed a micro-payment system for GSM calls, which uses hash-chain tokens as e-currency. They introduced a TTP, which is actively present in the flow of the system. Their proposition was that there was no need for public key operations on client side to provide an undeniable and secure payment scheme. However their assumption was falsified by Li, Wang, Zhou and Chen [31]. In [31] it is shown that a dishonest mobile user could make calls without payment and also a misbehaving service provider could overcharge for its service. The authors propose a new billing scheme in [31] and in their scheme TTP generates two signatures for a service, which are *Starting Time* Signature and *Finishing Time* Signature. The duration between these signatures determines the duration of the GSM call. There exist 5 communication steps and 4 public key operations. A limitation of the system is the billing scheme adds extra time to the call and makes the client pay more without getting service.

In [33], the authors use a high-level approach for billing and propose an architecture. Their focus is mostly its performance on a threshold based bandwidth management algorithm. Untraceability is not supported in the system. In [34], the authors propose UPASS; a double hash chain based prepaid billing architecture for WMNs. Their trust model is based on both classical certificate-based public-key cryptography and identity-based cryptography. The drawbacks of [33] are the complex trust and payment structures, missing simulative and/or analytical performance model, and disregarding users' anonymity/privacy. Similarly, UPASS does not consider client anonymity and untraceability.

Qi, Zhao, Wang and Choi [35] have introduced a security authentication and an undeniable billing protocol for WMNs. The proposed scheme uses RSA public key encryption for authentication purposes and elements of hash chains as e-currency. The proposed authentication protocol has key generation steps, which slows down the system. Furthermore the system does not fully provide anonymity and untraceability.

Many research on billing systems for WMNs have been done and many challenges have been tackled using cryptographic techniques. The general situation of the proposed systems is a lack of anonymity and untraceability support or bad performance for a high frequency payment scheme. SSPayWMN offers stable and affordable performance for payments with high frequency (needs to be verified by TTP too often) and also secure authentication with anonymity and untraceability properties.

2.4. DIFFERENCES AND SIMILARITIES WITH AN EARLIER WORK

The idea of SSPayWMN was first found and rooted by Can Yücel, in his thesis [8]. The motivation behind our thesis is same with [8], to build a secure and seamless micropayment system for wireless mesh networks. A basic system is taken over and improved.

The topology of the network used is analogous to previous network topology. System flow is very similar except some changes have been made. The packet flow used was from access points to operators, with mesh backbone and gateways in-between. The connection mediums between clients to the gateways were not encrypted and insecure lines. Gateways and operators were connected with secure links. The assumption of secure lines between gateways and operators is not changed but we also brought symmetric cryptography based security between gateways and access points. New system has become securer.

Most of the protocols have been changed; moreover we have added two new protocols, i.e. *Distribution of Public Keys* and *Change Alias*. However *Access Point Authentication* protocol is used intact as it appears in [8].

The main addition to the system is a Trusted Third Party (TTP), which brings an ultimate authority to the system. The usage of TTP and its servers provide credibility. Operators settle by communicating with TTP. Firstly clients pre-pay to TTP for the Internet service they are going to receive. Operators receive their payment from the TTP as they show the logs of their service.

[8] used operators as the end point of end-to-end protocols. We use TTP as the end point in order to provide undeniable verification and validation for critical applications.

In [8] some system entities were assumed to have public keys but an algorithm to distribute these keys were lacking. Existence of a TTP brings a possibility to use certificates in the system. SSPayWMN employs certificates for distribution of public keys. The distributed public keys could be broadcasted since they are signed by the TTP. This mechanism is added to SSPayWMN in this thesis.

Previous version of SSPayWMN did not clarify how the access points of the operators will be placed in the metropolitan area and the distribution of access points between operators. In SSPayWMN simulations two operators exist. Operators share the area in-between and compete to serve the users with stronger access points. There is a rivalry between operators since clients connect to stronger access points. An operator with low amount of investment for the system could not survive since high amount of access points are needed to serve in a wide range. The clients would not connect to an access point with a low signal rate if there is another access point with high signal rate in their range. As it will be explained later, there is no difference between operators in the sense of payment for the clients.

Anonymity and untraceability of the clients were not provided in [8]. These properties of the system should be provided by the system as long as there is no request for user actions and identity by a formal authority such as the police department. Current version of SSPayWMN provides anonymity and untraceability to some extent. Using aliases as client identity provides anonymity in the system. The aliases are changed periodically therefore only the actions between the periods can be tracked. The time period and detailed design will be explained in Section 3.2.

In [8], roaming between operators was costly therefore it was not 100% seamless to the client. Clients were customers of a particular operator, which implies an overcharge in the case of roaming to another operator. Current version of SSPayWMN makes every user of the system a client to the TTP. The TTP pays operators for their services. The new setting enables SSPayWMN to provide pure seamless roaming. In return *Seamless Mobility in Home Operator* and *Roaming* protocols have been changed to Seamless Mobility and *Seamless Roaming* respectively.

Internet service providing is simplified in the current version. The pre-payment was for a total packet size. In our version, clients buy hash tokens that provide Internet service for a predefined period of time. Seamless micropayment is preserved. Furthermore, we have increased the Update Packets interval in the system. In SSPayWMN, we set the update packets interval to a value slightly larger than $2\ x\ Token\ Renewal\ Period$. In a situation where a client that does not send the next hash token within this period, that means the client is dropped. Therefore the burden on the system caused by update packets is reduced significantly and with the new settings it is easier to handle the dropped clients.

In [8], simulations of the proposed scheme were done considering the same type of client. In this thesis, we have divided the clients into groups considering their socio-economical status. Speed, traveling distances, client's frequency of system usage are all affected by the client types and deterministic random distributions. Earlier version of SSPayWMN was lacking simulations of real-life situations, which are covered in the current version. Possible situations in real-life such as the diversity in mobility or system usage in time were not properly covered in [8]. In [8], burst scenario only covered users trying to authenticate themselves into the system on the same time. A rush hour scenario was lacking in which clients try to authenticate themselves in close time periods, but not necessarily on the same exact moment. Current version covered most of the possible scenarios within an ordinary day.

[8] employed the Random Way Walk Model [36] of the network simulator. In this model, clients' mobility patterns were totally random, with no logical base. These mobility patterns were not systematic, which causes unrealistic results in simulations. In this thesis, clients move from one point to another for a purpose. The new setting brings more realistic simulation results. The new network simulations are designed considering a real metropolitan area. We have used a model similar to Manhattan Mobility Model [37]. The setting of the roads is a grid, and the clients move from one location to another by using these roads. We have replaced the movement probabilities of the original Manhattan Mobility Model with random speeds and destinations. Speed and distances to destination are random but client types also affect them. Additionally our simulations cover a larger area with more access points than the simulations in [8]. Simulations in [8] had 32 access points with 16 gateways. Our simulations have 100 access points with 32 gateways.

A very significant improvement on [8] is the change of the network simulator used to simulate the system. In [8], the system was simulated using the Discrete Event Simulator: OMNET++ [38]. OMNET++ has offered GUI support and strong network simulation tools but it was lacking IEEE 802.11s support. Therefore, an ad hoc network simulation was executed to mimic the simulation of a real wireless mesh network. Converting the previous simulation results to a new network simulator was not feasible because there was no connection between OMNET++ and another network simulator. It was inevitable to implement the system from scratch on another network simulator, which has IEEE 802.11s wireless mesh network support. We have chosen network simulator 3 (ns-3) [39]. ns-3 supports IEEE 802.11s. However ns-3 did not support internetworking for mesh networks. It was infeasible for us to implement full internetworking and bridging functionalities for mesh networks on ns-3. As a solution, we have implemented virtual bridges between network nodes, and write every packet sent or received on text files. Every node in the system checks for packets to send in the text files. Every node in the mesh backbone had two interfaces in our design. The delay of passing the packets from one interface to another was neglected. This is, actually how internetworking and bridging functionalities of wireless mesh networks was mimicked.

Detailed and more realistic simulation settings and scenarios brought higher quality results. In this thesis, we focused on latency metrics for both independent protocol performances and for overall system performance. In [8], some latency tests have been done, but this did not reflect the typical behavior.

To conclude, in this thesis we have considerably improved [8] from both design and implementation (simulation) aspects.

3. REQUIREMENTS AND PROTOCOLS OF THE SYSTEM

In this section we will explain the requirements of the system and general system design. Then we will give brief explanation for system protocols. Finally we will explain the settlement of the protocols and money flow in the system.

3.1. REQUIREMENTS OF SSPAYWMN

A secure prepayment scheme for WMNs requires following attributes:

- **Wide Coverage:** Users should be getting service within a large area.
- **Seamless Roaming:** Users should connect and maintain their connection and continue to get service even while they are moving. Designed connection method should apply to different operators. From payment point of view, clients should be able to switch between operators as they move without a service interruption.
- **Seamless Mobility (Handoff):** Users should be able to switch between access points of the same operator as they move without a noticeable delay.
- **Anonymity:** It should not be feasible to track down a user's network actions from their payments (unless law enforcement requires doing so).
- **Mutual Authentication:** For preventing malicious use of network, both user and network should be mutually authenticated.
- **Two-way honesty:** Clients cannot deny that they did not take service. Operators cannot claim that they provide service more than they actually provide. These are to be guaranteed by using strong cryptographic protocols.
- **Untraceability:** It must not be possible to relate connection sessions of the users with other connection sessions. In this way, higher level of privacy could be provided.
- **Performance:** System should work effectively with stable performance. Additional caused by the system should be minimal.

3.1.1. General Design of the Network

Our SSPayWMN system does not only consist of a mesh backbone, but also Wi-Fi clients and wired servers. Mesh backbone basically relays the packets from clients to server so that the clients are able to get service.

Servers of the operators are wired and will be communicated via regular 802.3 Ethernet protocol in its local area. Mesh backbone will communicate within itself using IEEE 802.11s protocol. Clients will use IEEE 802.11a/b/g Wi-Fi protocols to connect to the access points/mesh routers.

3.1.2. General Overview of the Proposed Scheme

The proposed system supports user identification, authentication as well as authorization and accounting. The main objective is to design and develop a secure payment infrastructure for WMNs that also considers users' privacy and fairness. Our system model assumes the existence of mobile clients and operators, who will be charging the service they give. The operator's mesh backbone is made of several mesh routers, which are actually Access Points (APs) with IEEE 802.11s support. This backbone is connected to operator's server via a gateway. There exists a TTP (Trusted Third Party), which is reachable through an operator. These system components are listed together with their icons, used in the protocol figures, in Table 3.1.

Table 3.1. System Entities

	Mobile user (client)
	Access Point (AP) with mesh routing capability. From now on in this document, it is called as AP, but please note that it also has routing capability.
	Mesh backbone of the operator
	Gateway (GW) that connects the mesh backbone to outer world and also to the operator's server
	Operator's server (OP). Keeps necessary logs and user info.
	Trusted Third Party (TTP). Payment related logs are mostly to be generated by the TTP.

Since the clients are mobile, they may handover among different mesh routers (i.e. access points) of the same operators. They may also roam among different operators, not only due to coverage reasons, but also for having a better quality service. Our system aims to have seamless mobility (handoff) and seamless roaming for payment purposes such that when the client gets service through a new AP or switch to another operator, authentication and authorization are not performed from scratch.

From security point of view, we aim to have mutual authentication between client and the network in our protocols. Anonymity of the clients and untraceability across different usage periods (a.k.a. unlinkability) are privacy related goals of the protocols.

From payment point of view, our main aim is to have a fair system in which all the claimed transactions bear cryptographic proofs. In this way, the clients cannot repudiate using a service and the operators cannot claim for services that they do not provide. The latter is especially important during inter-operator settlement; it is also important to resolve client disputes.

The symbols used in this document are given in Table 3.2.

Table 3.2: The List of the Symbols

$\oplus$	XOR operation
$\|$	Concatenation
$E_K(X)$	Encryption of Xusing the key K
$D_K(X)$	Decryption of X using the key K
$h^n(X)$	Taking hash of Xn times
$HMAC_K(X)$	Taking HMAC of X using the key K
H_i	ith element of the hash chain (usage order)
$PU\text{-}TTP$	Public key of TTP
$PR\text{-}TTP$	Private key of TTP

AP_i	i^{th} Access Point or its identity
OP_i	i^{th} Operator or its identity
$PU\text{-}AP_i$	Public key of AP_i
$PR\text{-}AP_i$	Private key of AP_i
SN	Serial Number
N_X	Nonce created by entity X
PA	Previous Alias
NA	New Alias
$cert_i$	Public key certificate of AP_i
IV	Initialization Vector
TS	Timestamp
CR	Connection Request
DR	Disconnection Request
RR	Roaming Request
CAR	Change Alias Request
$MobReq$	Mobility Request
RP	Response (used in various protocol as positive acknowledgment)
DA	Disconnection Acknowledgement
$RAck$	Roaming Acknowledgement
$MobResp$	Mobility Response
$LASC$	Length of Anonymized Subhash Chain
HRI	Hash Token Renewal Interval
ASC	Anonymized Subhash Chain

3.1.3. Network Topology and General System Design

SSPayWMN employs previously explained system entities. The system entities are assumed to be located in a metropolitan area. While access points establish a mesh backbone and wait for clients to connect to them, gateways transmit the packets received from the access points to servers of the operators.

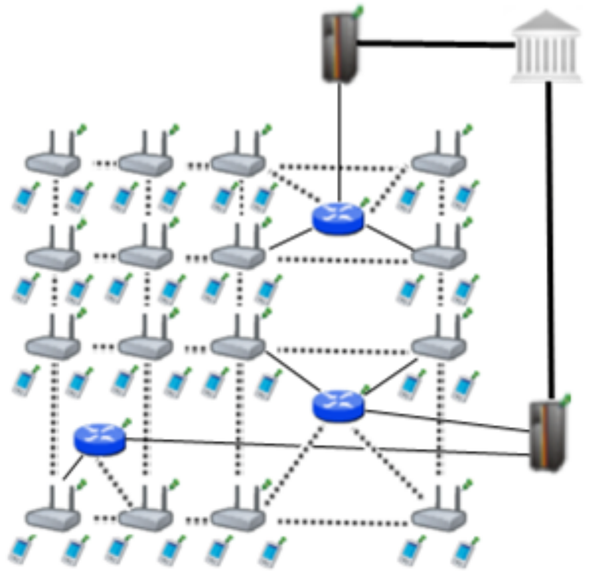

Figure 3.1. Network Topology

Figure 3.1 shows the topology of the network and connections between entities. Connection between serving access points is wireless and they use IEEE 802.11b/g Wi-Fi protocol. Mesh backbone uses IEEE 802.11s. The mesh backbone emulates a cloud from the mobile user's perspective. It is a black box; which receives packets from mobile user and delivers them to the gateway in a multi-hop manner. Mesh backbone uses Hybrid Wireless Mesh Protocol (HWMP) [40].

Connection medium between mesh backbone and gateway (GW) is either wireless or wired. GWs and operators communicate through wired connection. The connection between an operator and TTP is also wired. These connections use 802.3(Ethernet protocol).

3.1.4. Connection Card Structure

Connection Card is the main deed that clients buy from the TTP and use to get Internet service. We use a prepaid system, in which connection cards include tokens for credit generation. Please note that the tokens in the connection card are not directly used to pay for the Internet service, but to generate credits to pay for the Internet service. Hash tokens are generated using hash chains as discussed below. Connection cards also have unique *Serial Numbers* (SN), which are to be used for alias computation.

Tokens are basically links in a hash chain. For each set of tokens, the TTP picks on a random $seed$ and takes hashes of it many times. The number of hash operations is actually the number of token in a set. For example, if the client wants a hundred hash tokens, then the hash of $seed$ is taken hundred times. More formally a hash chain with 100 tokens is constructed in the following way.

$$H_0 = h(H_1) = h^{99}(seed)$$

$$H_1 = h(H_2) = h^{98}(seed)$$

$$H_2 = h(H_3) = h^{97}(seed)$$

$$\dots$$

$$H_{98} = h(H_{99}) = h^2(seed)$$

$$H_{99} = h(seed)$$

H_0 is the first token in the chain. The client uses this hash token to form a connection request for TTP. The generation of credits is explained in the following section.

Connection Cards are refillable with hash tokens, which are to be sold by the TTP. Operators compete with each other to provide high-quality service for broadband access in the WMN since the users are assumed to have free roaming.

Serial Number SN is a 128-bit value. With this setting, the system is able to support up to 2^{128} users. Hash tokens are to be generated using SHA-256 hash algorithm; hence they are 256-bit long.

Considering current technology, smart cards are suitable tools to be connection cards. A simple Connection Card with 4 KB memory could store a SN and approximately 1000 hash tokens.

3.1.5. Anonymized Hash Chains

Clients change their aliases periodically to make their actions unlikable to their aliases. However, an adversary could trace a client's actions by tracing the link between the hash tokens of the client.

To provide untraceability in the system, clients form up anonymized subhash chains from the original hash chain. The client and the TTP determines the amount of the hash tokens that will be used in the next session. The Change Alias Interval (CAI) and Hash Token Renewal Interval (HRI) determine the Length of Anonymized Subhash Chains ($LASC$) as following:

$$Length\ of\ Anonymized\ Subhash\ Chains\ (LASC) = CAI/HRI$$

In unit simulations and real-life scenario simulations we have used $CAI = 60$ and $HRI = 5$.

An anonymized subhash chain for an original hash chain is calculated as explained below.

Recall that an original hash chain with n elements is denoted as follows.

$Original\ hash\ chain$: $\{H_i, H_{i+1}, H_{i+2}, H_{i+3}, H_{i+4}, H_{i+5}, \ldots\ldots, H_n\}$

When the client wants to use the hash token H_i to form a connection request, she has to form an anonymized subhash chain. The last token of the anonymized subhash chain is calculated as follows:

$$H'_{i+LASC-1} = h(H_{i+LASC-1} \oplus Nonce)$$

Client and the TTP calculate the $LASC$ elements of the anonymized subhash chain by performing hash operations on $H'_{i+LASC-1}$ as follows:

$$Anonymized\ Subhash\ Chain\ (ASC) \\ = \{h^{LASC}(H'_{i+LASC-1}), h^{LASC-1}(H'_{i+LASC-1}), \ldots, H'_{i+LASC-1}\}$$

A sample for the generation of the anonymized subhash chains is depicted in Figure 3.2.

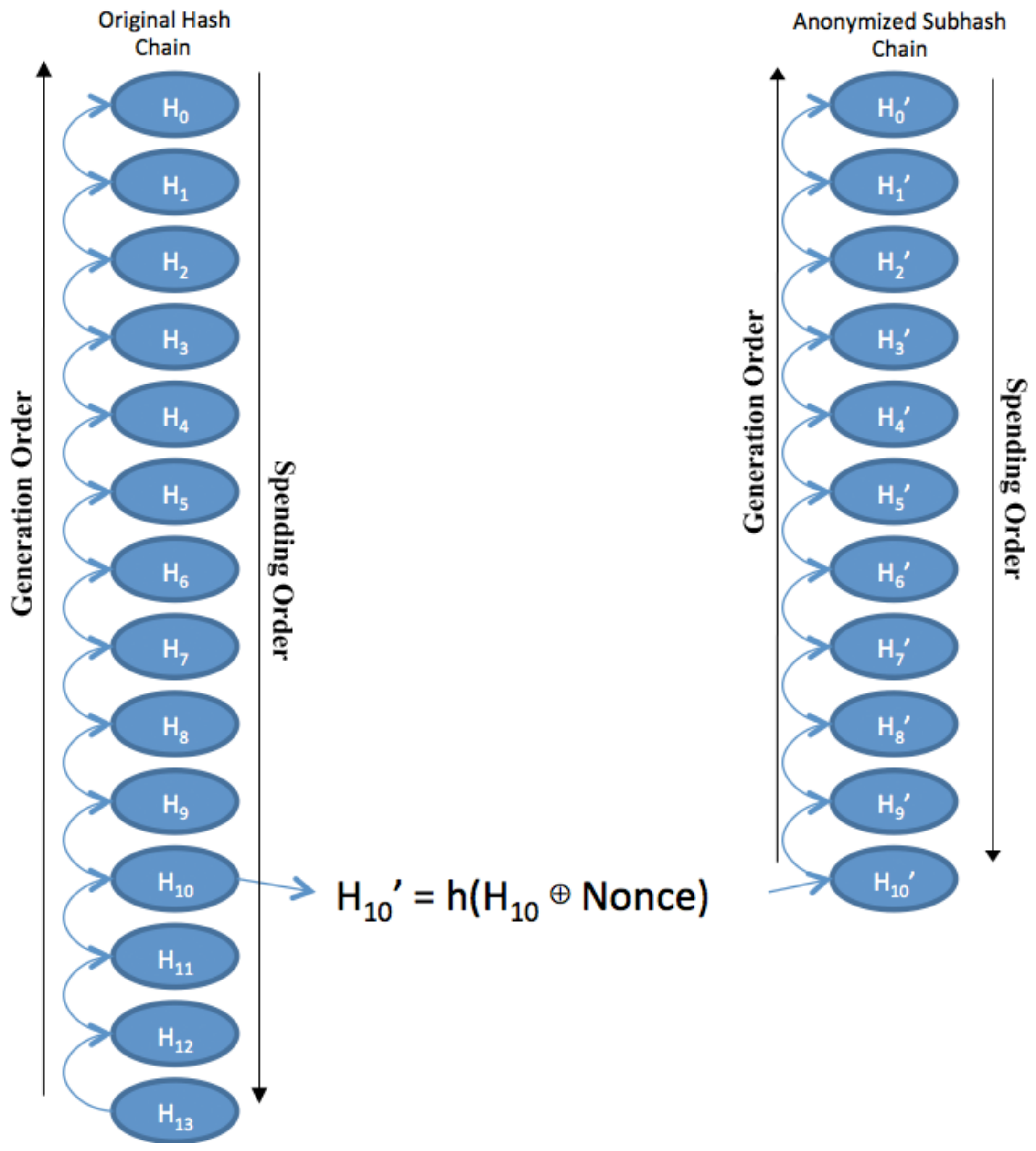

Figure 3.2. A Sample for Generation of Anonymized Subhash Chains with $CAI = 55$, $HRI = 5$ and $LASC = 55/5 = 11$

Before any authentication or change alias phase, the client sends the first hash token of the remaining hash chain (In Figure 3.2 the first hash token is H_0). TTP knows the *seed* value of the client. TTP and the client are able to form the anonymized subhash chain simultaneously. When the client sends the first hash token of the remaining hash chain to the TTP, TTP counts backwards from the received hash token $LASC$ times. TTP computes the corresponding anonymized hash token and takes the hash of the output $LASC$ times and calculates $ASC = \{h^{LASC}(H'_{i+LASC-1}), h^{LASC-1}(H'_{i+LASC} - 1), \ldots, H'_{i+LASC-1}\}$. These operations form up an anonymized subhash chain and the anonymized hash tokens are spent in reverse order as shown below.

$$First\ token\ to\ use = h^{LASC}(H'_{i+LASC-1}) = H'_i$$

$$Second\ token\ to\ use = h^{LASC-1}(H'_{i+LASC-1})\ verified\ as:$$

$$h(H'_i) == h^{LASC}(H'_{i+LASC-1}) = H'_{i+1}$$

$$\ldots$$

$$Last\ token\ to\ use = H'_{i+LASC-1}\ verified\ as:$$

$$h(H'_{i+LASC-1}) == H'_{i+LASC-2}$$

In a case of a disconnection or connection drop before spending all the hash tokens in the anonymized subhash chain, client stores the index of the last used hash token index. For the next connection request the client sends the first hash token in the remaining hash chain to the TTP. For the new session, both the client and the TTP generate a new anonymized subhash chain.

In a mobility situation, clients transfer the next anonymized hash token to the new access point. The clients start to get service from the new access point when the transfer is finished.

An adversary could not relate two different anonymized subhash chains since using the hash output of XOR of a hash token and a random nonce value generates the hash chains. Every time a new anonymized subhash chain is generated a different nonce value is used. The hash operation on the seed of the anonymized subhash chain prevents any relation between different anonymized subhash chains. More formally, consider the following two anonymized subhash chain seed calculations:

$$Anonymized\ Subhash\ Chain\ Seed_1(ASHS_1) = h(H_i \oplus Nonce1)$$

$$Anonymized\ Subhash\ Chain\ Seed_2(ASHS_2) = h(H_j \oplus Nonce2)$$

Here suppose H_i and H_j are on the same hash chain, for an adversary it is infeasible to find out $H_i \oplus Nonce1$ and $H_j \oplus Nonce2$ by using $ASHS_1$ or $ASHS_2$. Therefore, an adversary could not discover any hash token in the original hash chain by exploiting anonymized subhash chain because of the irreversibility property of hash algorithms. Thus, it cannot link H_i and H_j. Therefore, we can say that usage of anonymized subhash chains provides unlinkability among different sessions.

3.1.6. Alias Computation

Aliases are temporary identifiers for clients. They change frequently using a secure protocol. Untraceability is achieved by changing aliases by the previously stated way however it is only durable to some extent.

The serial number (SN) of the Connection Card, which is bought from an operator, will be used as a base for client's aliases. An alias will be computed by performing the following operations:

1. Client will pick a random 128-bit unsigned number and call it her $Nonce$.
2. Perform XOR operation with SN and her nonce; take the hash of the output. $h\,(SN\ \oplus Nonce)\ = Alias$
3. Client will use this alias whenever her identity is required.

Aliases are 128-bit values; even if it is a very small possibility to have the same alias with another client at a given point of time, there is still a nonzero probability. To address this problem, TTP checks the proposed alias to be a unique one. This check is done in *Change Alias* protocol, which will be mentioned in Section 3.2.

The nonce values used in computation of the aliases are to be sent in encrypted messages to the TTP in the related protocol. Therefore only the client and the TTP can relate the aliases originated from a particular SN.

3.2. PROTOCOLS OF THE SYSTEM

In this section, we will explain the protocols of the system. There are 10 protocols in the system. End-to-end Two-way protocols consist of Initial Authorization and *Reuse of a*

Connection Card, *Disconnection* and *Change Alias* protocols. Initial Authorization and *Reuse of a Connection Card* protocols are executed each time a client connects to the system. These protocols are not necessary for ongoing connections. *Change Alias* protocol exists to provide untraceability to some extent. Clients' actions could not be related to previous sessions when they execute *Change Alias* protocol. *Disconnection* protocol is necessary for operators to collect their money from the TTP. *Disconnection* protocol marks the end of the provided service. End-to-end Two-way protocols send packets from clients to the TTP and return the packets back to clients. *Update Packets* protocol is an End-to-end One-way protocol, which sends packets from clients to the TTP. *Update Packets* protocol ensures the stability of the system and prevents any inconvenience in a case of connection interruption. Other protocols i.e. *Seamless Mobility*, *Seamless Roaming*, *Packet Transfer* and *Access Point Authentication* are independent protocols and do not belong to any group. *Seamless Mobility* and *Seamless Roaming* protocols are executed in mobility situations of the clients. *Distribution of Access Point Public Keys* protocol is performed for distributing the signed public keys of the access points. This protocol is not implemented in the simulations and it is assumed to occur before the system deployment.

3.2.1. Initial Authorization and Reuse of a Connection Card

Initial Authorization is the beginning for system usage. Whenever a client purchases new hash tokens from the TTP, she will need to authorize herself to TTP. Initial Authorization Protocol, shown in Figure 3.3, achieves mutual authentication and authorization of the user.

The clients may disconnect before using up all the credits in a connection card. *Reuse of a Connection Card* (*Reuse-CC*) protocol allows the clients to connect using the remaining credits in a card. *Reuse-CC* protocol does not differ extensively from *Initial Authorization* protocol. The main difference is instead of sending first hash token; the client sends whichever token is the next one. Alias will change before the protocol starts. Both protocols compute new aliases before sending the Connection Requests (CR). The crucial point here is that TTP should be able to update last hash value entry of the client in the database and associate it with the new alias.

The access point is a member of a mesh backbone and a particular access point is to be selected according to its transmission power. Since it is assumed that all access points have the same attributes, the serving access point is the closest access point to the client.

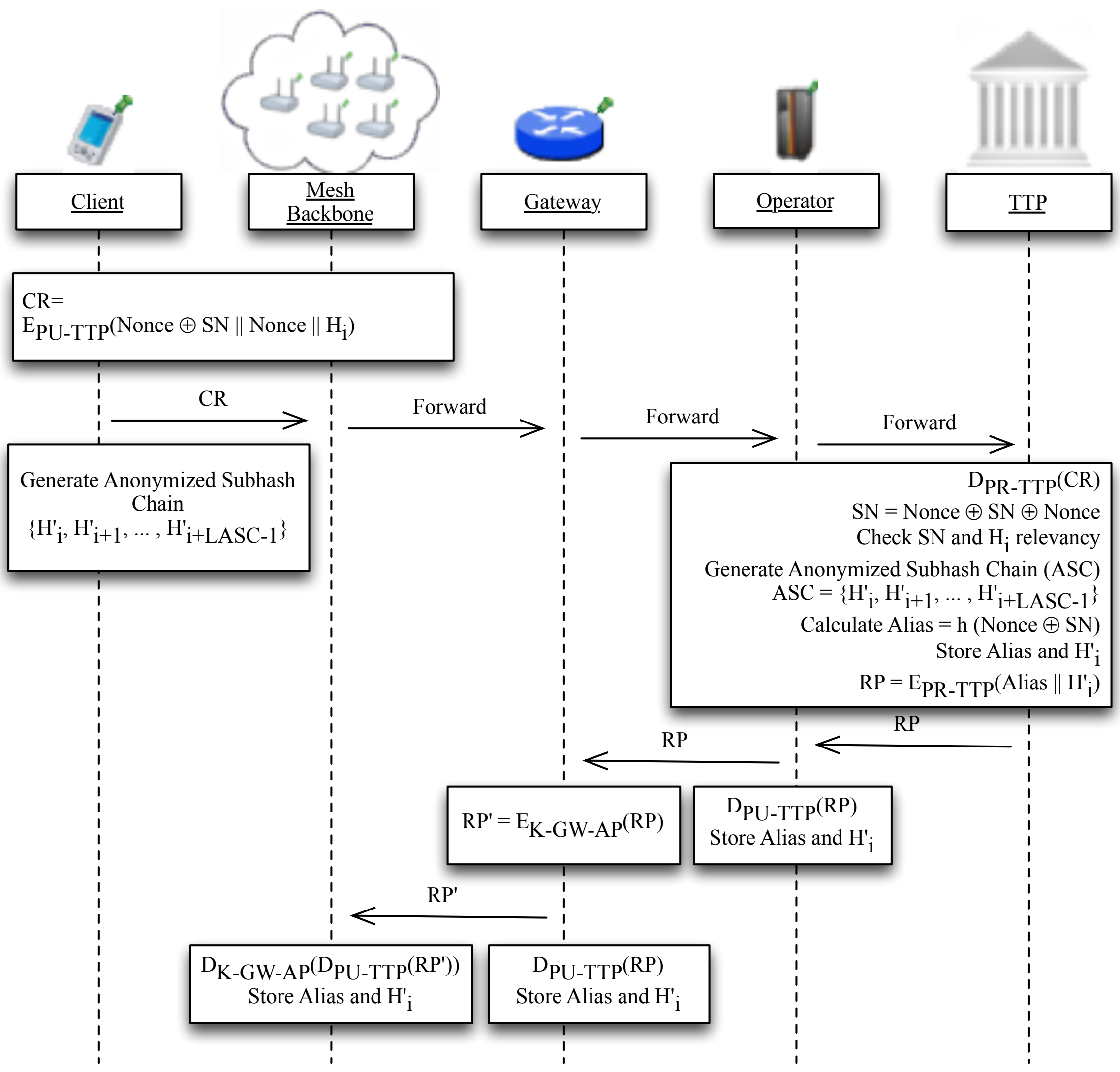
Client
Mesh Backbone
Gateway
Operator
TTP
CR= E_PU-TTP(Nonce ⊕ SN || Nonce || H_i)
CR
Forward
Forward
Forward
Generate Anonymized Subhash Chain {H'_i, H'_i+1, ... , H'_i+LASC-1}
D_PR-TTP(CR)
SN = Nonce ⊕ SN ⊕ Nonce
Check SN and H_i relevancy
Generate Anonymized Subhash Chain (ASC)
ASC = {H'_i, H'_i+1, ... , H'_i+LASC-1}
Calculate Alias = h (Nonce ⊕ SN)
Store Alias and H'_i
RP = E_PR-TTP(Alias || H'_i)
RP
RP
RP' = E_K-GW-AP(RP)
D_PU-TTP(RP)
Store Alias and H'_i
RP'
D_K-GW-AP(D_PU-TTP(RP'))
Store Alias and H'_i
D_PU-TTP(RP)
Store Alias and H'_i

Figure 3.3. Initial Authorization and Reuse-CC

Mobile clients introduce themselves to the operator using *Initial Authorization* protocol. $H_i = H_0$ in *Initial Authorization* protocol, $H_i = H_k$ $k > 0$ in *Reuse of a Connection Card* protocol. TTP already knows mobile user's serial number (SN) and the first element, H_0, of her hash chain. The mobile user does not want to reveal her SN to any adversary because that SN will be used all the time and it is sensitive information from security and privacy points of view. To achieve anonymity, the mobile client computes an alias and uses this value instead of SN. The mobile client will change her alias periodically as she continues to get service (*Change Alias* protocol will be explained later).

Initial Authorization and *Reuse-CC* steps are described below.

1. Client computes an alias using a nonce $Nonce$ that she generated.
   a. $Alias = Nonce \oplus SN$
   b. $H_i = h^{T-i}(IV)$ (The CC is assumed to have T credits)
   c. $CR = E_{PU-TTP}\ (Nonce \oplus SN \parallel Nonce \parallel H_i)$
   d. Client sends this CR to AP_S.
   e. Client starts to generate an anonymized subhash chain as the network processes client's CR.
2. AP_S receives the connection request and relays the request through mesh backbone.
3. Gateway receives the CR and relays it to the operator.
4. Operator relays CR to TTP.
5. TTP receives the connection request (CR) and decrypts it using its private key.
   a. $D_{PR-TTP}\ (Nonce \oplus SN \parallel Nonce \parallel H_i) = Nonce \oplus SN \parallel Nonce \parallel H_i$
   b. TTP checks alias' uniqueness within its database of users, it would make the client start over the protocol if alias is not unique.
   c. It computes $Nonce \oplus SN \oplus Nonce = SN$.
   d. TTP checks SN and H_i association. Store $Nonce \oplus SN$ and H_i
   e. TTP computes $Anonymized\ Seed = h\ (H_{i+LASC-1} \oplus Nonce)$
   f. TTP generates anonymized subhash chain by taking the hash of $Anonymized\ Seed\ LASC$ times. The output of the last hash operation gives the first token of the anonymized subhash chain, which is H'_i.
   g. TTP computes $RP = E_{PR-TTP}\ (Alias \parallel H'_i)$
   h. TTP sends RP to the Operator.
6. Operator receives RP and verifies the signature using public key of TTP.
   a. Operator sends RP to the gateway.
   b. The Operator gets $Alias$ and H'_i and stores these values.
7. GW receives RP and verifies the signature using public key of TTP.
   a. GW uses the shared secret key with AP_S and calculates $RP' = E_{K-GW-AP}(RP)$
   b. GW sends RP' to the AP_S.

   c. GW verifies the signature of TTP.
   d. GW stores $Alias$ and H'_i.
8. AP_S receives RP' and decrypts it using the shared secret key with GW.
   a. AP_S verifies the signature using public key of TTP.
   b. It calculates $Alias$ and H'_i and stores these values.

The wired links are secured however the medium between GW and APs are insecure; therefore, the packets that are sent through this medium are encrypted with shared secret keys between GWs and APs.

3.2.2. Access Point Authentication

After authentication processes of the client with the TTP, a second authentication step begins. Client and access point will mutually authenticate each other for safe communication; this protocol ensures the feature -Mutual Authentication- of SSPayWMN.

Figure 3.4, describes the protocol briefly.

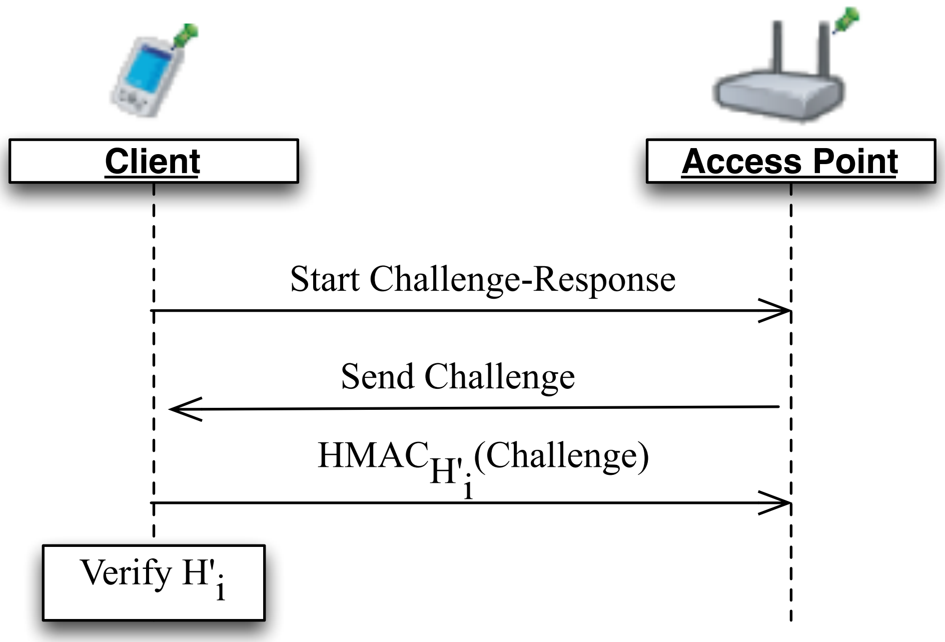

Figure 3.4. Access Point Authentication

1. AP_S sends a challenge request to the client, which started connection.

2. When client receives this challenge request:
    (a) Client drops the packet if it is not the AP_S that she sent connection request.
    (b) Client drops the packet if there was not any CR.

   If (a) and (b) are 3 invalid then the client sends a 128-bit challenge to the AP_S.
3. AP_S takes the HMAC of this challenge, and uses relevant anonymized subhash token as the key of HMAC.
    (a) $Response = HMAC_{H'_i}(Challenge)$
    (b) AP_S sends $Response$ to the client.
4. Client also takes the HMAC of the challenge and uses the stored anonymized subhash token H'_i as the key. Then it compares the result with the one that access point sent. If it is authenticated, client starts to use access point to get Internet service.

3.2.3. Packet Transfer

After mutual authentication of client and AP_S, the client starts to send data packets as shown in Figure 3.5. These packets represent the Internet usage of the client. Client could send as many packets as she wants in the period of time, which is bought by a hash token. In this protocol the usage of hash tokens and data packets are explained.

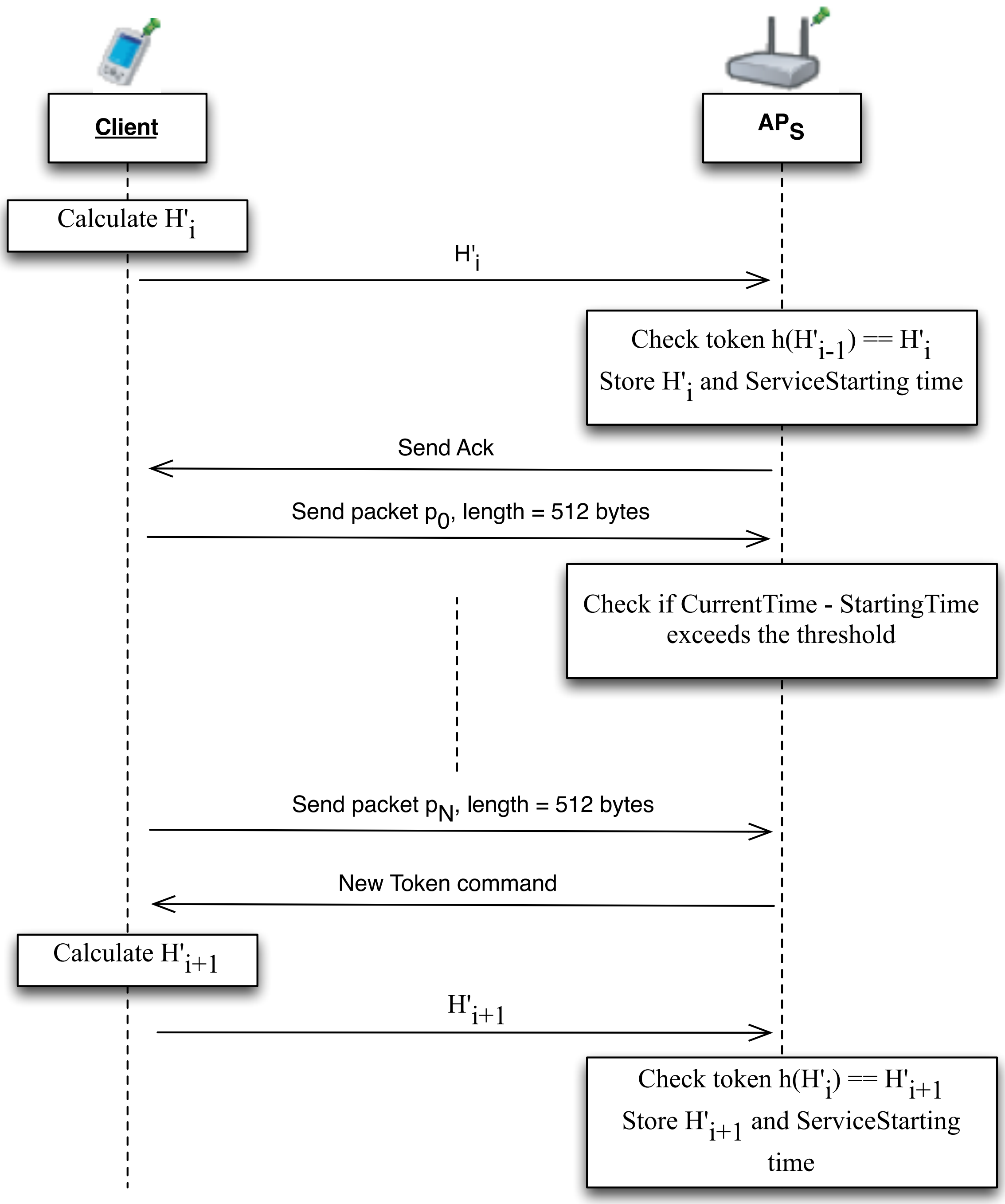
Client
AP_S
Calculate H'_i
H'_i
Check token h(H'_{i-1}) == H'_i
Store H'_i and ServiceStarting time
Send Ack
Send packet p_0, length = 512 bytes
Check if CurrentTime - StartingTime exceeds the threshold
Send packet p_N, length = 512 bytes
New Token command
Calculate H'_{i+1}
H'_{i+1}
Check token h(H'_i) == H'_{i+1}
Store H'_{i+1} and ServiceStarting time

Figure 3.5. Packet Transfer

1. Client starts the session with the first anonymized hash token (in this case current has value is H'_i) of the remaining hash chain.
2. AP_S receives H'_i, and updates client's service starting time.
   (a) Checks if $h(H'_{i-1}) == H'_i$

(b) If true sends acknowledgement (Ack) to client and updates currently used hash value as H'_i.

3. Client sends first 512-byte data packet p_0.
4. If the client gets served for over the threshold value (5 minute interval is used in simulations) then the AP asks for the next hash token.
5. The steps between (1) and (4) are repeated as long as client gets Internet service.

3.2.4. Changing Alias

Anonymity property is achieved by using aliases, but complicated part is achieving untraceability. The aliases should change on a basis that an adversary, who knows a certain client's alias, could not be able to trace client's activity on her home network, and also could not trace her movements among the operators or access points. Clients generate anonymized subhash chains to break the link between the hash tokens that they receive service. Clients do not fully trust to the operator and its access points and assume that the access points could store and link the hash tokens of different sessions.

To be able to change alias in a safe way, client needs to communicate with TTP but interrupting TTP very often would slow down the entire operation due to extra delays. Therefore periodic changes of aliases are mandatory and these changes are achieved by making access points to ask all of the active clients for new aliases after a certain period of time. Attackers or access points themselves would know that aliases are changed but would not know the mapping between old aliases and the new ones. Such a protocol is also used in Mix Networks [41].

Simultaneous alias changes aim to prevent attacks that would aim to analyze network traffic of access points and examine connection requests. Enforcing alias change by the access points, a more generalized control over the clients is achieved. Attackers could not understand which client wanted to change her alias, because all the clients getting service from a particular access point have requested to change their aliases at that particular time.

The client should request changing alias, because client and the TTP should be the only parties who know association between an alias and a client's SN.

$Alias\ Change\ Timer$ is a local timer that runs on every Access Point. All of the timers are set roughly to the same time manually. System designer decides on the time value on which the access point will count down from (50 minutes of time period is used in simulations). The timer period is updateable by the TTP. TTP knows every access points' public key, it could send new interval by encrypting the new value with the public keys of the access points. However, this process is not covered in simulations.

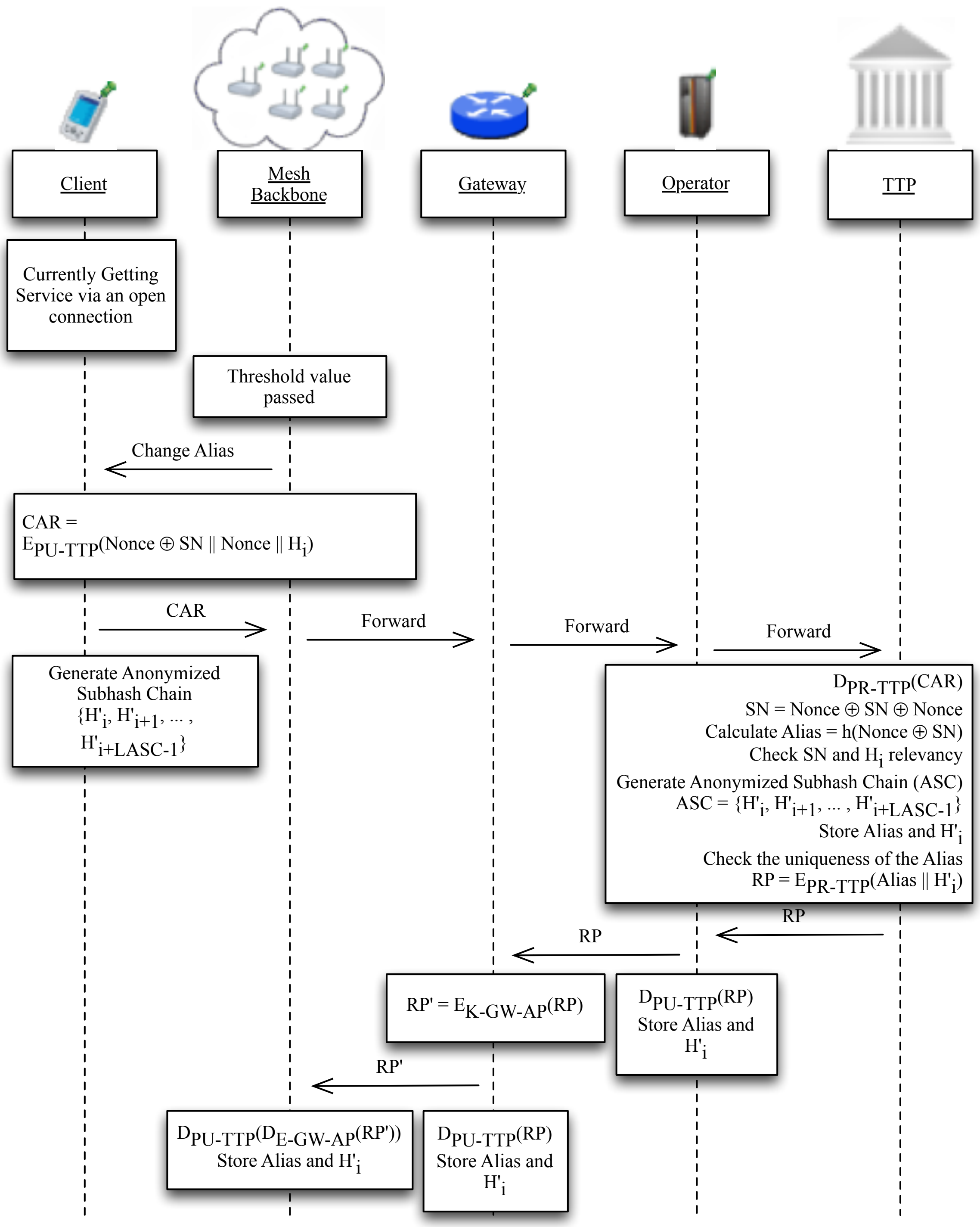
Client
Mesh Backbone
Gateway
Operator
TTP
Currently Getting Service via an open connection
Threshold value passed
Change Alias
CAR = $E_{PU\text{-}TTP}(Nonce \oplus SN \,||\, Nonce \,||\, H_i)$
CAR
Forward
Forward
Forward
Generate Anonymized Subhash Chain $\{H'_i, H'_{i+1}, \ldots, H'_{i+LASC-1}\}$
$D_{PR\text{-}TTP}(CAR)$
$SN = Nonce \oplus SN \oplus Nonce$
Calculate Alias = $h(Nonce \oplus SN)$
Check SN and H_i relevancy
Generate Anonymized Subhash Chain (ASC)
$ASC = \{H'_i, H'_{i+1}, \ldots, H'_{i+LASC-1}\}$
Store Alias and H'_i
Check the uniqueness of the Alias
$RP = E_{PR\text{-}TTP}(Alias \,||\, H'_i)$
RP
RP
$RP' = E_{K\text{-}GW\text{-}AP}(RP)$
$D_{PU\text{-}TTP}(RP)$ Store Alias and H'_i
RP'
$D_{PU\text{-}TTP}(D_{E\text{-}GW\text{-}AP}(RP'))$ Store Alias and H'_i
$D_{PU\text{-}TTP}(RP)$ Store Alias and H'_i

Figure 3.6. Changing Alias

Change Alias protocol is shown in Figure 3.6 and described below.

1. Client continues to get service, in other words uses the *Packet Transfer* protocol.

When the $Alias\ Change\ Timer$ countdown finishes, Access Points broadcast "Change Alias" command to all of their clients. The interval value is a system parameter; 50 minutes of interval value is used in the simulations.

2. Client receives "Change Alias" command.
    a. Client computes a new alias by picking up a new random nonce $Nonce$ and computes $Nonce \oplus SN$.
    b. Client forms a Change Alias Request (CAR)
    c. $CAR = E_{PU-TTP}(Nonce \oplus SN \parallel Nonce \parallel H_i)$
    d. The client sends the CAR to AP_S.
    e. Client generates a new anonymized subhash chain as the TTP processes the connection request of the client.
3. AP_S receives CAR and relays it to the GW via mesh backbone.
4. Gateway forwards CAR to operator.
5. Operator forwards CAR to TTP.
6. TTP receives Change Alias Request (CAR) and decrypts it using its private key.
    a. $D_{PR-TTP}(Nonce \oplus SN \parallel Nonce \parallel H_i) = Nonce \oplus SN \parallel Nonce \parallel H_i$
    b. TTP checks for new alias' $h\ (Nonce \oplus SN)$ uniqueness and starts over the protocol if not unique.
    c. TTP computes $Nonce \oplus SN \oplus Nonce = SN$.
    d. TTP computes $Anonymized\ Seed = h\ (H_{i+LASC-1} \oplus Nonce)$
    e. TTP generates anonymized subhash chain by taking the hash of $Anonymized\ Seed$, $LASC$ times. The output of the last hash operation gives the first token of the anonymized subhash chain, which is $H'_i = h^{LASC}\ (H_{i+LASC-1} \oplus Nonce)$.
    f. It checks SN and H_i association and stores $Alias = h\ (Nonce \oplus SN)$ and H'_i.
    g. It computes $RP = E_{PR-TTP}(Alias \parallel H'_i.)$.
    h. TTP sends RP to operator.
7. Operator receives RP.
    a. Operator sends RP to the GW.
    b. The operator computes $Alias$ and H'_i and stores these values.
8. GW receives RP.

   a. The GW encrypts the RP and calculates $RP' = E_{K-GW-AP}(RP)$
   b. GW sends RP' to the AP_S.
   c. The GW calculates $Alias$ and H'_i, and stores these values.
9. AP_S receives RP' and decrypts it as follows:
   a. $D_{K-GW-AP}(RP')$
   b. The AP_S verifies the signature using public key of TTP.
   c. The AP_S reveals $Alias$ and $H'_i$and stores these values.

3.2.5. Update Packets

In standard flow of the system, after authentication, access points handle the accounting. Because of the fact that access points keep the last alias and token of the client they are able to validate next token by performing hash operation to the token they kept and compare it with new coming hash token. However it is essential to send periodic updates to the TTP to provide stability in the system in the case of client drops.

Access points keep track of ongoing communications, after some time passed without update from a user it send disconnection request by itself. When access points broadcast change alias commands they delete all the record related to previous connections therefore they do not send unnecessary disconnection packets to TTP.

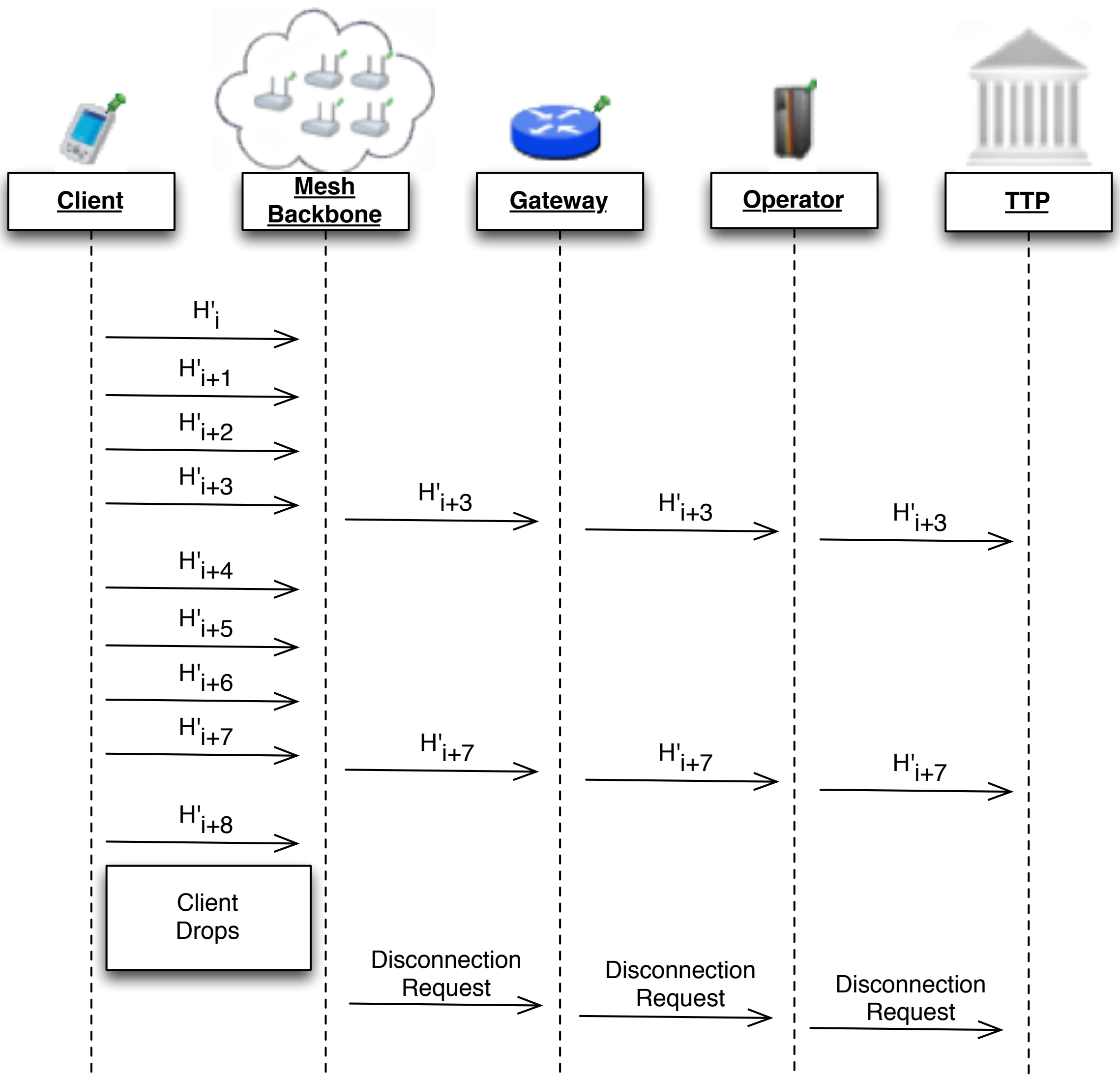

Figure 3.7. Update Packets

Protocol design of Update Packets protocol is shown in Figure 3.7 and the details of the protocol are explained below.

1. After client sends the first anonymized subhash token, the access point starts to count the time passed. After t units of time (value of t is a system parameter, 1 minutes of an time interval is used in simulations), access point encrypts the Alias and lastly used hash token using the public key of the TTP and sends this cipher text to the GW.
2. The GW receives the update packet and forwards it to TTP through related operator.
3. TTP receives the update packet and decrypts the packet using its private key. TTP updates the last token used by the client.

4. In a case of client drops from the network, access point concatenates the Alias, hash value and a time stamp and encrypts them with the public key of TTP. Sends it to TTP as a disconnection request from the client.

3.2.6. Disconnection

To be able to run Reuse-CC, the client has to run a proper disconnection protocol. The Update Packets protocol brings stability to the system in case of a connection interruption, but the main assumption is that most of the users will be disconnecting from the operator using the *Disconnection* protocol that we explain in this section and in Figure 3.8.

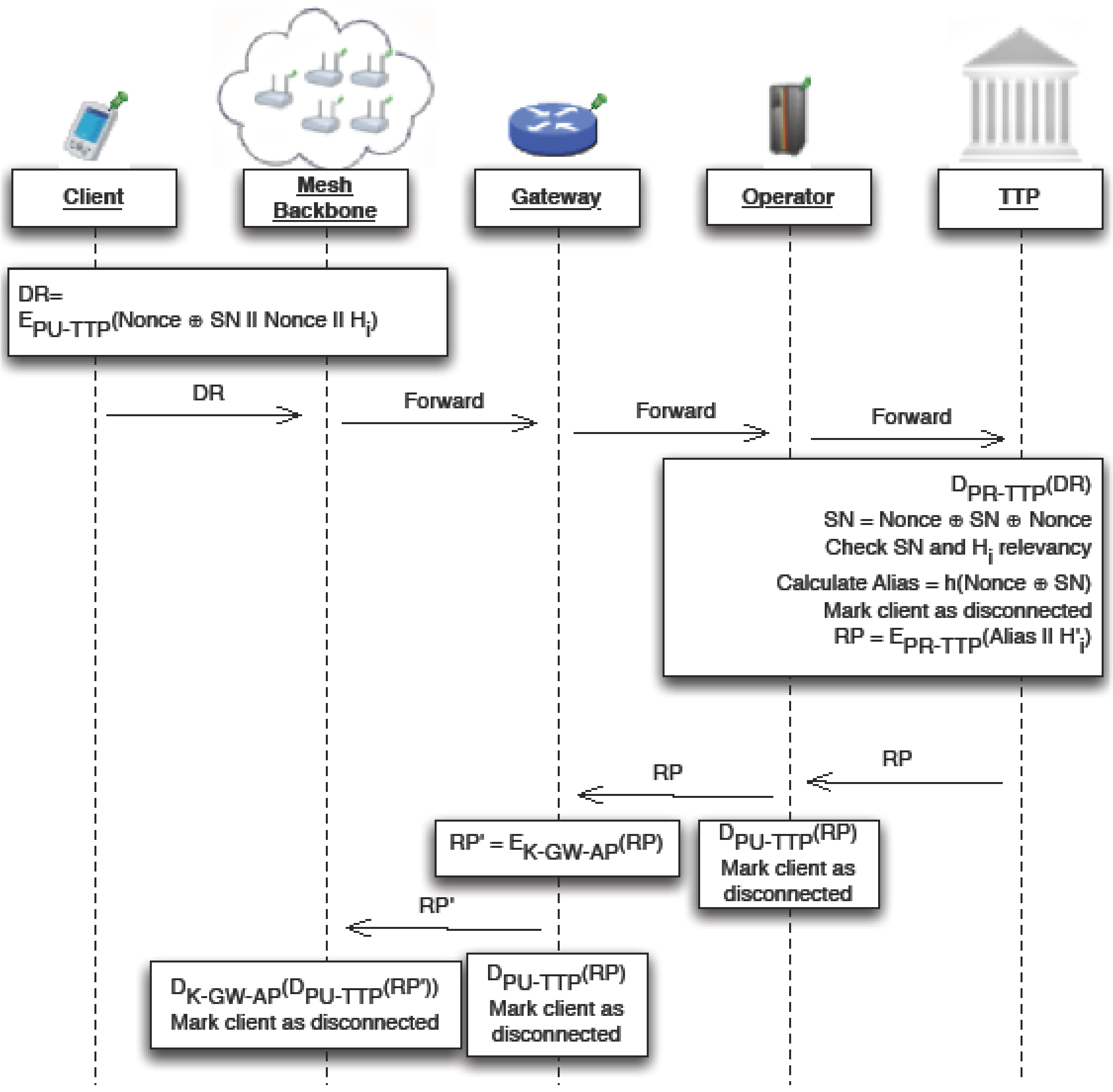
Client
Mesh Backbone
Gateway
Operator
TTP
DR= E_{PU-TTP}(Nonce ⊕ SN II Nonce II H_i)
DR
Forward
Forward
Forward
D_{PR-TTP}(DR)
SN = Nonce ⊕ SN ⊕ Nonce
Check SN and H_i relevancy
Calculate Alias = h(Nonce ⊕ SN)
Mark client as disconnected
RP = E_{PR-TTP}(Alias II H'_i)
RP
RP
RP' = $E_{K-GW-AP}$(RP)
D_{PU-TTP}(RP) Mark client as disconnected
RP'
$D_{K-GW-AP}$(D_{PU-TTP}(RP')) Mark client as disconnected
D_{PU-TTP}(RP) Mark client as disconnected

Figure 3.8. Disconnection

Disconnection protocol is described below.

1. Client forms a disconnection request
   - $DR = E_{PU-TTP}\ (Nonce \oplus SN \parallel Nonce \parallel H_i)$
   - Client sends the packet to the AP_S.
2. AP_S relays DR to the mesh backbone, to make it reach to the GW.
3. GW receives and forwards the DR to the Operator.
4. Operator receives and forwards the DR to the TTP.
5. TTP receives the DR.
   - TTP calculates SN and checks the relevancy between SN and H_i.
   - TTP computes the Alias, similar with the previous end-to-end two-way protocols.
   - TTP computes $Disconnection\ Acknowledgement\ DA = E_{PR-TTP}\ (Alias \parallel H'_i)$
   - TTP sends the DA to the Operator.
6. Operator receives DA.
   - Operator relays DA to GW.
   - Operator verifies the signature and marks client as disconnected.
7. GW receives DA.
   - It relays DA to the mesh backbone.
   - GW verifies the signature and marks client as disconnected.
8. AP_S eventually gets the DA, verifies the signature on it and disconnects the particular client, which corresponds to the $Alias$ it received. Ideally access points are assumed to delete all information about the past connections for the sake of freeing memory space. However if operators decide to trace user's actions then they could do so for a limited time until the client changes its $Alias$.

3.2.7. Distributing Access Point Public Keys

Achieving seamless mobility in home operator and also to support seamless roaming, a public key distribution mechanism is integrated in SSPayWMN system.

In Figure 3.9, a generic model for public key distribution is shown. This protocol has two parts; one is certificate generation for access point public keys, the other one is distribution of the public keys. The part between operator and the TTP is offline. This part of the protocol runs during set-up, before the deployment of the access points in the field.

If an operator wants to add a new access points to the metropolitan area then it should perform the same protocol but his time only for the new access points.

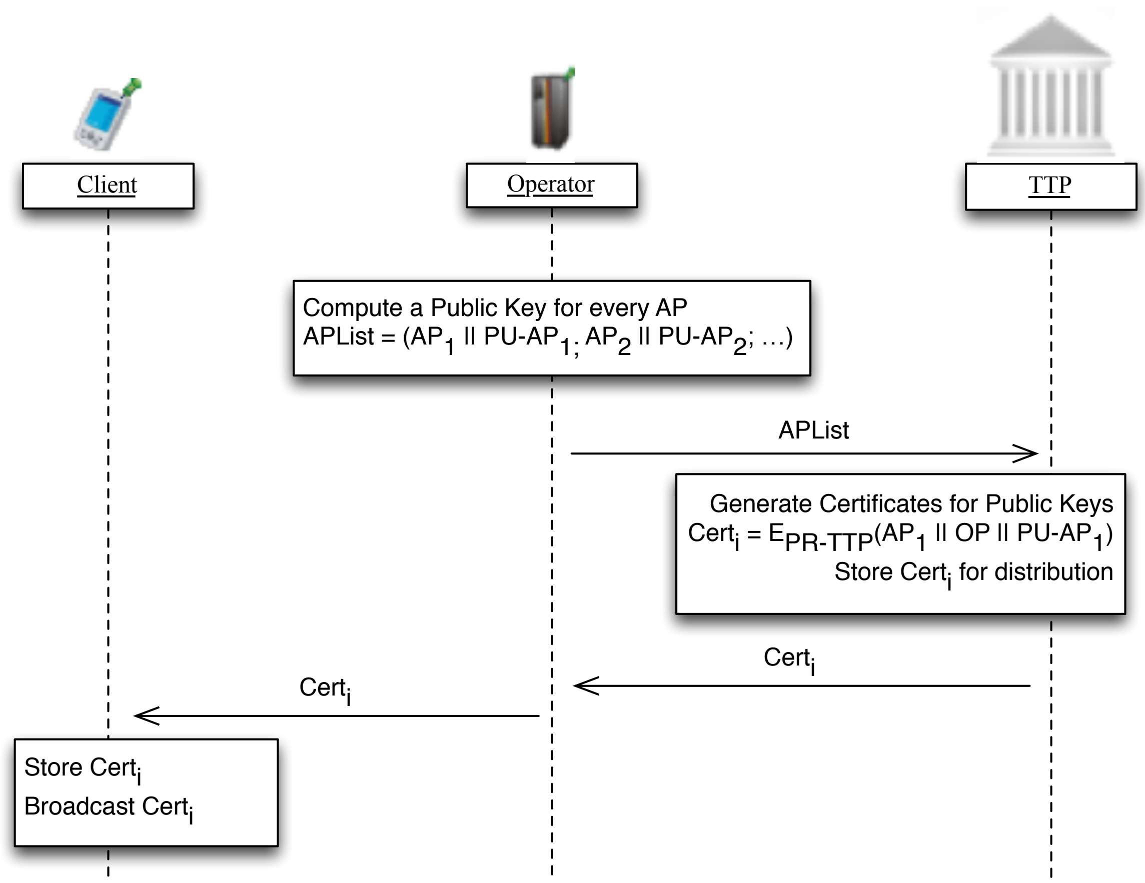
Client
Operator
TTP
Compute a Public Key for every AP
APList = (AP1 || PU-AP1; AP2 || PU-AP2; ...)
APList
Generate Certificates for Public Keys
Certi = EPR-TTP(AP1 || OP || PU-AP1)
Store Certi for distribution
Certi
Certi
Store Certi
Broadcast Certi

Figure 3.9. Distributing Access Point Public Keys

Distributing Access Point Public Keys algorithm is described below.

1. Operator generates public/private key pairs for the access points in its mesh backbone and embeds these keys to them before the deployment.
    - Operator forms an access point list ($APList$); which consists of access points and their corresponding public keys.

    - Operator sends this list to the TTP through a secure channel or in offline manner.
2. TTP receives the $APList$ and starts to generate certificates for every access point and public key pair.
    - Certificates are formed as:
    - $Cert_i = E_{PR\text{-}TTP}(AP_i \parallel OP \parallel PU\text{-}AP_i)$
    - TTP stores these certificates for distribution.
    - Other protocols are employed (such as *Initial Authorization* or *Reuse-CC* protocols) of SSPayWMN for certificate distribution. Suppose an AP does not possess its certificate. In such a case whenever this access point gets a connection request it will concatenate a certificate request to the packet. When the TTP receives such a request, it concatenates corresponding certificate to the connection response. Then, TTP sends the connection response and $Cert_i$ together to the operator.
3. Operator receives the connection response and the certificate and relays these packets to the access point through gateway and mesh backbone.
4. Access point receives and stores its certificate and broadcasts it to the nearby access points.

3.2.8. Seamless Mobility in Home Operator

Seamless Mobility is run whenever the client performs a handover between two access points of her home operator.

Every access point has its public/private key pair and ability to broadcast its public key, handoff in home operator could be handled in a seamless way. As it is shown in Figure 3.10, client gets a signed handover ticket from its old access point and uses this signed ticket to maintain to get the Internet service from a new access point.

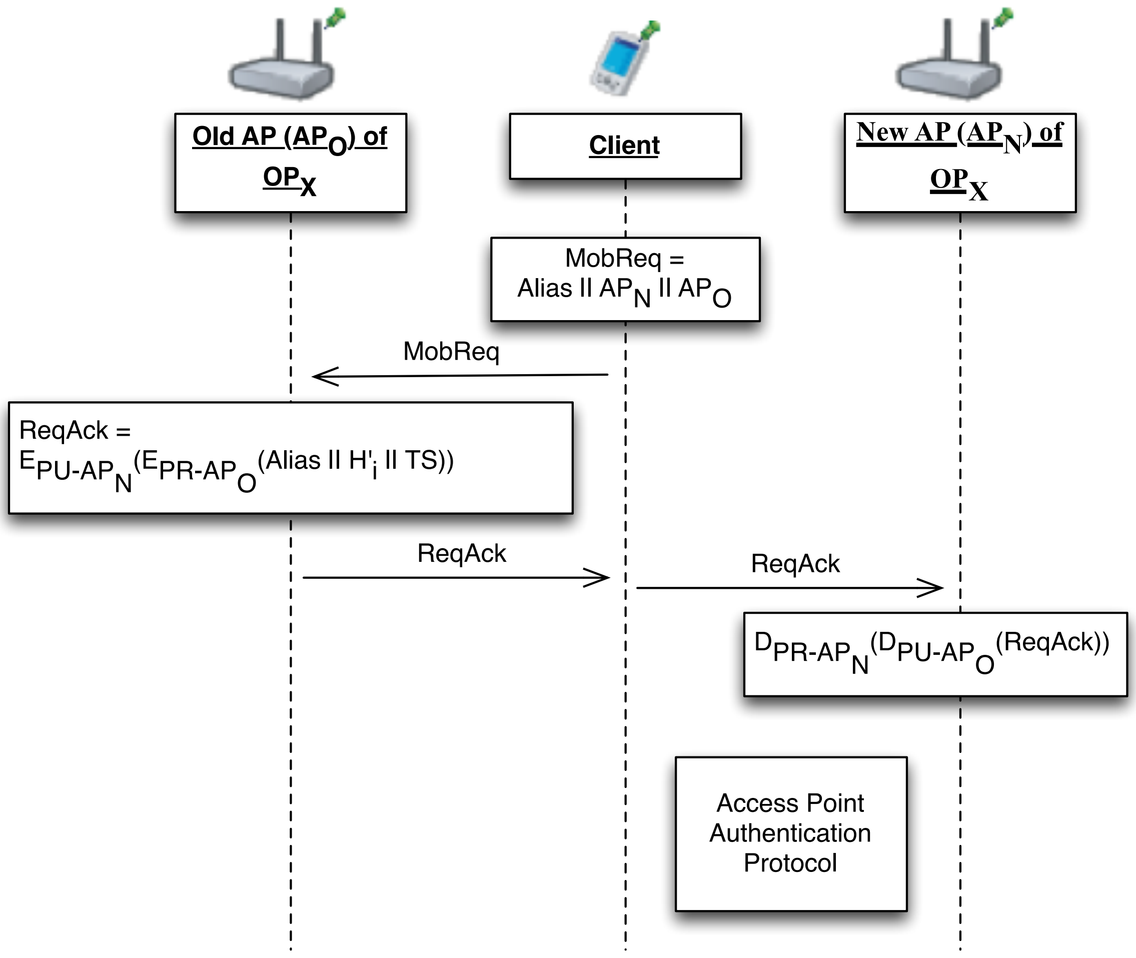
Old AP (AP_O) of OP_X
Client
New AP (AP_N) of OP_X
MobReq = Alias II AP_N II AP_O
MobReq
ReqAck = E_PU-AP_N(E_PR-AP_O(Alias II H'_i II TS))
ReqAck
ReqAck
D_PR-AP_N(D_PU-AP_O(ReqAck))
Access Point Authentication Protocol

Figure 3.10. Seamless Mobility in Home Operator

Seamless Mobility protocol is described below. In this setting, AP_O is the last access point that the client gets service. AP_N, is the access point that the client would like to continue to get service.

1. Client sends a Mobility Request ($MobReq$) to AP_O.
    - $MobReq = Alias \parallel AP_N \parallel OP_N$
2. AP_O receives $MobReq$ and forms a Request Acknowledgement ($ReqAck$).
    - $ReqAck = E_{PU-AP_N}(E_{PR-AP_O}(Alias \parallel H'_i \parallel TS))$
    - AP_O sends $ReqAck$ to the client.
    - $ReqAck$ consists of the mobility ticket that the client uses to get services from the AP_N. It is signed by AP_O and encrypted for AP_N.
3. Client receives $ReqAck$ and forwards it to the new access point (AP_N).

4. AP_N decrypts $ReqAck$ using its private key.
    - AP_N reveals the signed ticket of the AP_O. AP_N sends this signed data to its affiliated operator to use it for collecting funds from TTP.
    - AP_N verifies the signature over this signed ticket using AP_O's public key. Then, it checks TS in order to decide whether the ticket has expired or not.
    - Then, AP_N starts a challenge-response protocol with the client.
    - The rest of this protocol is the same as *Access Point Authentication* protocol.

3.2.9. Seamless Roaming

Seamless Roaming is run whenever the client changes the serving access point with a new one, and starts to get service in a different operator's network.

Public Key Cryptography enables us to handle seamless roaming without performing the authentication process from scratch. As it is shown in Figure 3.11, client gets a signed roaming ticket from its old access point and uses this signed ticket to maintain to get Internet service from a new access point. In parallel the old access point sends a disconnection request to its operator for the client.

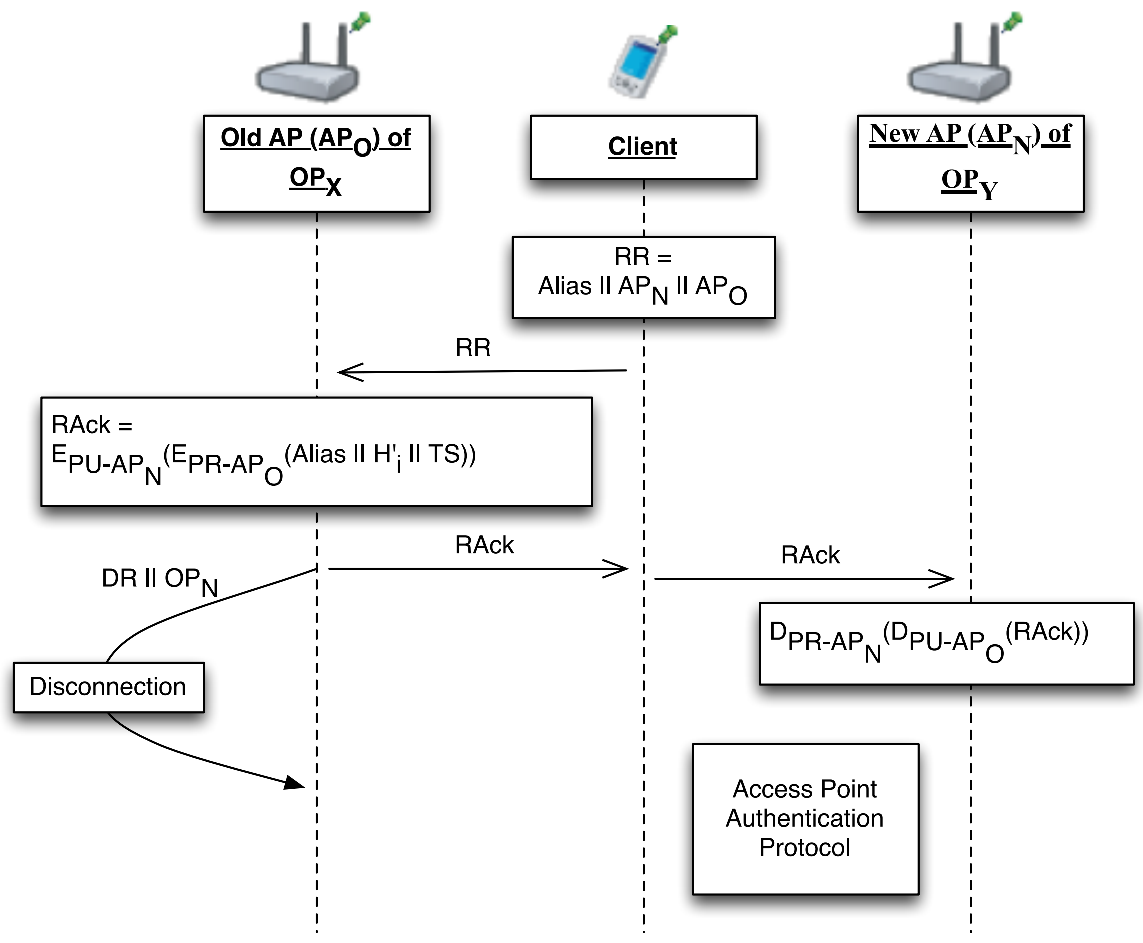
Old AP (AP_O) of OP_X
Client
New AP (AP_N) of OP_Y
RR = Alias II AP_N II AP_O
RR
RAck = E_PU-AP_N(E_PR-AP_O(Alias II H'_i II TS))
RAck
RAck
DR II OP_N
Disconnection
D_PR-AP_N(D_PU-AP_O(RAck))
Access Point Authentication Protocol

Figure 3.11. Seamless Roaming

Seamless Roaming protocol is described below. In this protocol, the client would like to switch from its old operator (OP_O) to a new one (OP_N). In this setting, AP_O is the last access point that the client gets services from OP_O. AP_N, is the access point that the client would like to continue to get services in OP_N network.

1. Client sends a $Roaming\ Request$ (RR) to AP_O.
   - $RR = Alias \parallel AP_N \parallel OP_N$
2. AP_O receives RR and forms a Roaming Acknowledgement ($RAck$).
   - $RAck = E_{PU-AP_N}(E_{PR-AP_O}(Alias \parallel H'_i \parallel TS))$
   - AP_O sends $RAck$ to the client.
   - $RAck$ consists of the roaming ticket that the client uses to get services from the AP_N.
3. AP_O starts the disconnection protocol for the client after sending the $RAck$.

- This disconnection protocol runs in parallel with the roaming protocol. Thus it does not put an extra delay in roaming. Old operator (OP_O) stores disconnection acknowledgement (DA) to support its claim to get funds for the services that it provided until roaming occurs. TTP stores the information that this disconnection is due to a roaming to OP_N in order not to get confused when AP_N disconnects without a connection request reached to it.
- In this scheme, AP_O's signed ticket serves as a formal document, which represents the beginning of the session with AP_N.

4. Client receives $RAck$ and forwards it to the new operator (AP_N).
5. AP_N decrypts $RAck$ using its private key. The rest of this step is the same with (4) of *Seamless Mobility* protocol.

3.2.10. Payment to the Operators (Settlement)

The settlement of the operators is similar with the settlement step in [8]. The purpose and the technique is the same, however the log design and TTP operations are different. In other words, settlement step of SSPayWMN is altered but not completely changed.

In SSPayWMN, operators claim their money from the TTP by showing their service logs. A log proves a service that has been provided between a connection request and a disconnection request.

$$Log = OpId \mid\mid Connection\ Request \mid\mid Signed\ Connection\ Response \mid\mid TS$$

Operators store connection requests (CR) of the clients; CRs are formed in the *Initial Authorization* and *Reuse of a Connection Card* protocols. When a client makes a DR, operator stores it. After receiving the DR, operator forms its log as follows.

$$Log = OpId \mid\mid Disconnection\ Request\ (DR) \mid\mid Signed\ Disconnection\ Response \mid\mid TS$$

TS stands for timestamp in the logs. TSs are mandatory in the logs to make TTP's job easier.

When TTP receives two consecutive logs from an operator:

1. TTP will sort the logs according to their TS value.
2. TTP first decrypts CR since it is encrypted with the public key of TTP. CR consists of $Alias$, $Nonce$ and the first hash token to be used to get service.

   Consider

   $$CR = E_{PU-TTP}\ (Nonce \oplus SN \parallel Nonce \parallel H_i)$$

   TTP decrypts it using its private key, and gets SN by the XOR operation:

   $$Nonce \oplus SN \oplus Nonce = SN$$

   Note that SN's first token used is H_i.

   $Connection\ Requests$ and $Disconnection\ Requests$ are formed by using the tokens from the original hash chains. Therefore, anonymized subhash tokens are not needed to be used in settlement phase.
3. TTP decrypts the Signed Connection Response using its public key, and gets the alias and the hash token. TTP compares the values with the ones in connection request. If they match, then the log is marked as valid.
4. The abovementioned log is only a service starter; operator needs to show service-ending log to claim its money from the TTP.

   Service ending log naturally has a larger TS value; therefore this log comes later in the sorted list of logs.

   TTP takes the ending log and decrypts DR using its private key.

   TTP gets $Alias$, $Nonce$ and the hash token from the decrypted DR. TTP makes the XOR operation: $Nonce \oplus SN \oplus Nonce = SN$ and gets the SN. Note that SN used is the hash token came with the DR to end the service.
5. TTP takes the Signed Disconnection Response and decrypts it using its public key. TTP gets the alias and the hash token from it, and compares the values with the ones came with the DR. If the values match, TTP considers the log as a valid service-ending log.
6. After validating the logs, TTP performs the hash operation over service ending hash token until it reaches the service starter hash token. TTP counts these hash operations. This count is mapped to funds for the provided service.

However the misusage of the logs should be addressed. Consider the situation of a client:

- Gets service from her home operator between H_0 and H_{10}
- Gets service from a foreign operator between H_{11} and H_{20}
- Gets service from her home operator between H_{21} and H_{30}

In this type of situation home operator has two CRs and DRs, whereas foreign operator has a CR and DR. Home operator has the following logs:

$$Log1 = OpID \,||\, CR_{H_0} \,||\, Signed\ RP_{H_0}$$

$$Log2 = OpID \,||\, DR_{H_{10}} \,||\, Signed\ DA_{H_{10}}$$

$$Log3 = OpID \,||\, CR_{H_{21}} \,||\, Signed\ RP_{H_{21}}$$

$$Log4 = OpID \,||\, DR_{H_{30}} \,||\, Signed\ DA_{H_{30}}$$

The home operator has served between H_0 and H_{10} and also has served between H_{21} and H_{30}. Home operator would want to take the money for serving between H_{11} and H_{20}. It could pretend that it has served the client between H_{11} and H_{20} by not sending $Log2$ and $Log3$. Since $Log2$ indicates that client is disconnected from the operator at H_{10} and $Log3$ suggests that the client started to get service from the operator at H_{21}. Sending only $Log1$ and $Log4$ results TTP to think that the home operator has served the client between H_0 and H_{30}. This way operator would want money for serving 30 hash tokens.

Abovementioned situation suggests that there should be another operator, which has served between H_{11} and H_{20}. Second operator would have two logs as follows.

$$Log5 = OpID \,||\, CR_{H_{11}} \,||\, Signed\ RP_{H_{11}}$$

$$Log6 = OpID \,||\, DR_{H_{20}} \,||\, Signed\ DA_{H_{20}}$$

Foreign operator proves that it has served between H_{11} and H_{20} by showing the signed RP and DA.

TTP would see that it has already paid home operator for service to that particular client between H_{11} and H_{20}. This means that home operator has tricked TTP to pay more.

In the proposed system TTP is the one who has the authority, it pays operators their money. If the TTP finds an operator misbehaving it could give a penalty to the operator and do not pay for future services, or there could be several other kinds of penalties, since TTP has the proof it could bring the subject to the court as well.

4. PERFORMANCE EVALUATION

In this section we will explain the simulation environment and properties of system entities in the implementation. Then timings of cryptographic operations on different system entities will be explained. Unit test results and comments on independent protocol performances will follow. Then we will present the client models and mobility scheme. The overview of real-life scenario simulation results will be given. Finally, every real-life scenario simulation result for protocols will be shown and explained.

4.1. SIMULATION ENVIRONMENT

The simulations of SSPayWMN are conducted using ns-3. ns-3 is a widely used, popular and a free network simulator. ns-3 is a discrete event network simulator. It is mainly used in research and academia. The ns-3 project is still being developed by ns-3 users, since it is an open platform for developers. ns-3 supports 802.11s and wireless mesh networks and coding is done in C++ programming language.

The simulator was run on a computer with 2.4 GHz Intel Core 2 Duo, 2 GB 1067 MHz DDR3, Apple MacBook OSX v10.6.8.

The network topology is hierarchical and WMN supports connections with other IEEE 802.11 protocols, clients communicate with TTP via access points, GWs and operators in sequence. Access points are connected to gateways with 6-54 Mbps Wi-Fi connection. Some important specifications about the access points are shown in Table 4.1. *Update Interval* determines the time value between two update packets that access point send to TTP.

Table 4.1: AP Specifications

AP-Gateway Connection bit rate	6-54 Mbps – Wi-Fi
AP-Gateway Distance	70 m
Service Duration per token	5 minutes
Update Interval	1 minutes

The network consists of 32 gateways and 100 access points. In unit simulation there is only one mobile client whereas in real-life scenario simulations there are 100-300-500 mobile clients.

4.1.1. Cryptographic Operations and Their Timings

Public Key Cryptography timings for access points and gateways are mentioned in [42]. For operator servers and TTP servers, timings from [43] are used. For mobile clients, performance values for iPhone 4 are computed using openssl [44] by running the command "openssl speed". The details of the results are given in Appendix 1.

Platform specifications are shown in Table 4.2, and RSA-2048 timings are shown in Table 4.3. For AES timings the values from [21] are used and they are shown in Table 4.4. Timings of hash algorithms are taken from [45] and they are presented in Table 4.5.

Table 4.2: Platform Specifications

	Gateway [42]	Linksys WRT54GS (AP) [42]	Server [43]	Client [44]
CPU Speed	2.08 GHz	200 MHz	Dual-core 64 bit 2.8 GHz	Not disclosed by Apple
CPU type	AMD Athlon XP 2800	Broadcom MIPS32	Intel Xeon	Arm Cortex-A8
RAM	512 MB	32 MB	-	-

Table 4.3: RSA-2048 Timings

	Gateway [42]	Linksys WRT54GS [42]	Server [43]	Client [44]
RSA Private Key Operation	47.3 ms	1529.0 ms	8.13 ms	120 ms
RSA Public Key Operation	1.3 ms	37.9 ms	0.32 ms	3.3 ms

Table 4.4: AES-128 Timings

	Gateway [21]	Linksys WRT54GS [21]	Server	Client
Approximate AES-128 Timings per block	0.001 ms	0.01 ms	-	-

Table 4.5: SHA-256 Timings

	Gateway	Linksys WRT54GS [45]	Server [45]	Client [44]
Approximate SHA-256 Timings per 256-bit block	-	0.02 ms	0.0002 ms	0.008 ms

4.2. UNIT TEST RESULTS

Unit tests are simulation runs per standalone protocol behavior. In these tests there is only one user, and this user performs the same protocol every minute. These tests are done to ensure the robust behavior of modules of the system.

As discussed in Section 3, some protocols show similarity considering packet sizes, cryptographic operations and packet routes. Since there would be no difference between unit tests of protocols that are in the same group, there is one result chart for a particular group of protocols.

4.2.1. Unit Test Result for End-to-End Two-Way Protocols

Unit tests for end-to-end two-way protocols consist of a user, running the same protocol every minute. End-to-end Two-way protocols consist of *Initial Authorization*, *Reuse-CC*, *Change Alias* and *Disconnection* protocols. Figure 4.1 presents the average delay of packet delivery over time. In this simulation the user sends the packet to a serving access point and the packet hops 2 times in the mesh backbone until it reaches the gateway. Gateway forwards the packet to operator and operator transmits the packet to TTP. TTP processes this packet and sends it back to the client through the same route.

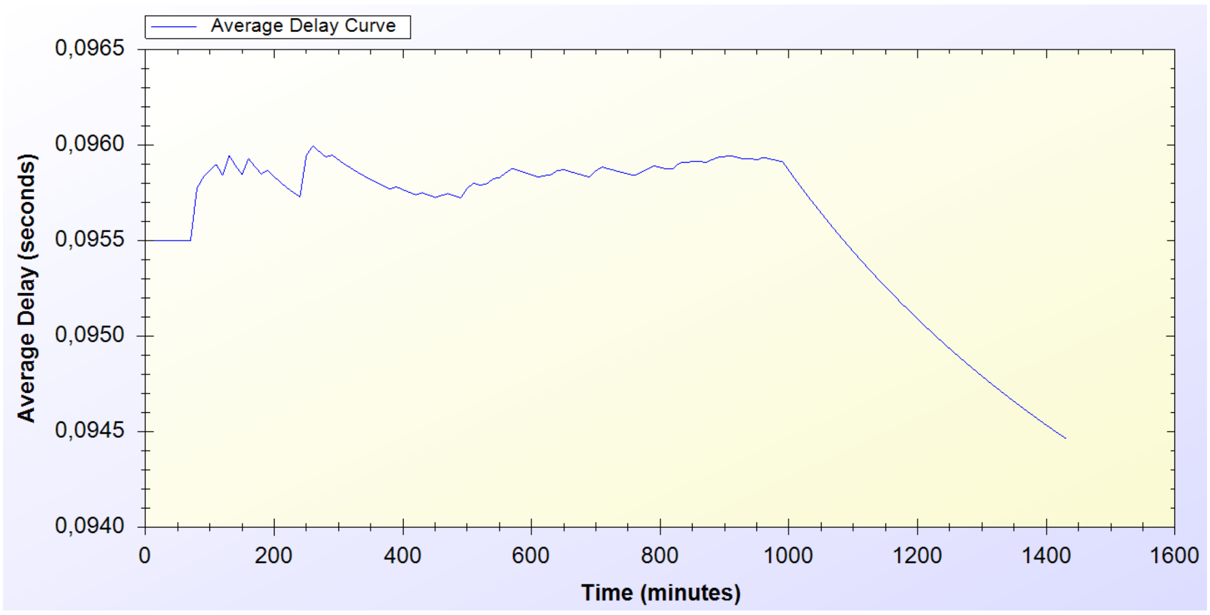

Figure 4.1. End-to-End Two-Way Protocols Unit Test Result

There is an initial delay that shows variation around 0.1 second. This unstable behavior is caused by different initial packet delays. System needs some packets to set up paths between mesh nodes. The performance stabilizes in time and shows a significant decrease by the end of the simulation. Average delay converges to 0.0945 second.

4.2.2. Unit Test Result for Access Point Authentication

Access Point Authentication protocol consists of a challenge-response protocol. It contains two HMAC operations.

Unit test for this protocol contains a user, trying to run access point authentication protocol with a serving access point every minute. The resulting chart, presented on Figure 4.2, shows the average delay of the protocol versus time.

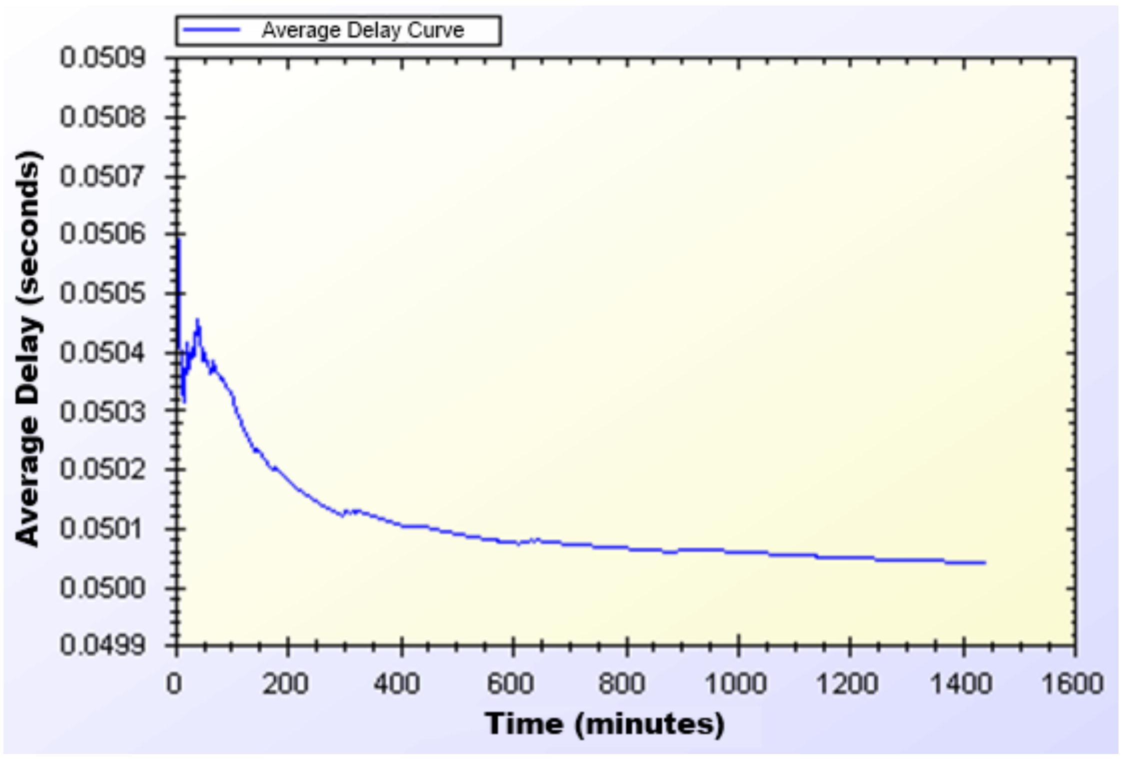

Figure 4.2. Access Point Authentication Protocol Unit Test Result

Average delay of access point authentication converges to 0.05 second in the steady state. The initial delay values are higher than the later ones, because nodes need some time to establish and see who is around. At the time of initial deployment, wireless nodes send and receive beacons and perform operations using them.

4.2.3. Unit Test Result for Seamless Mobility and Roaming

Seamless Mobility and *Seamless Roaming* protocols have the same behavior since client sends and receives same length of packets. Thus, they are grouped together for unit tests.

Unit test for *Seamless Mobility* and *Seamless Roaming* protocols consist of a client changes serving access point every minute. Client is located in between two access points and these access points are both eligible for service. Since these protocols must be seamless to the user it is important to get reasonable delays for these protocols.

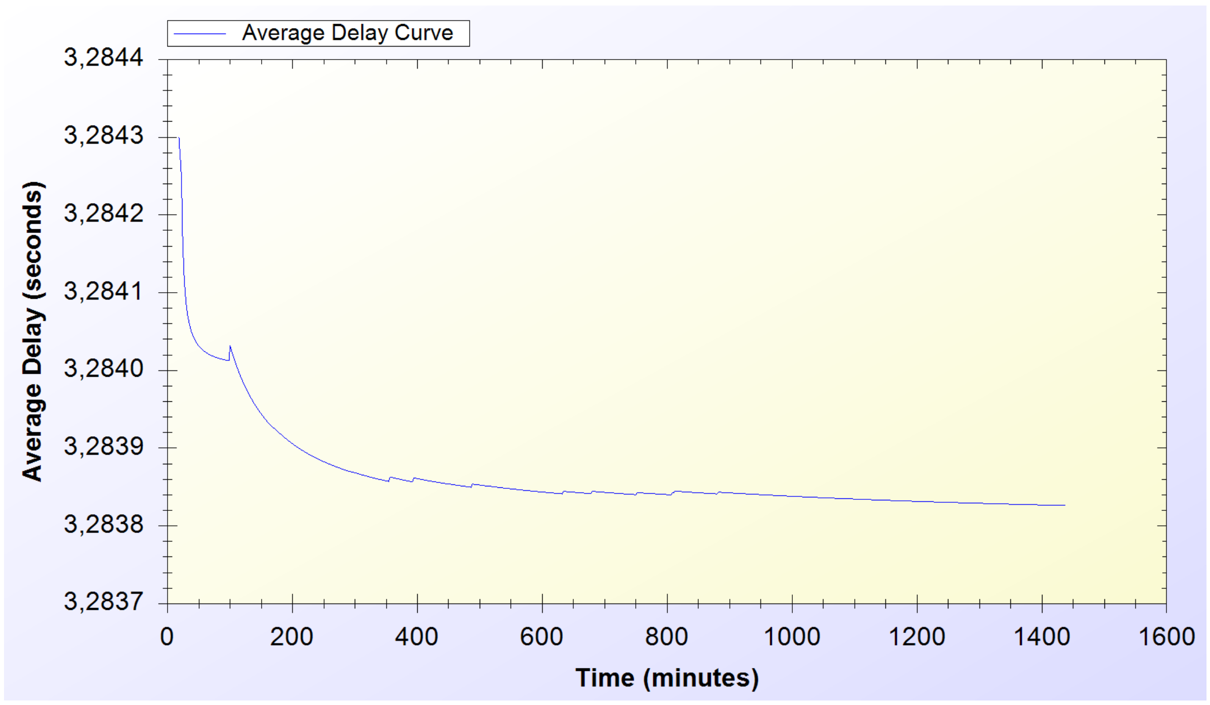

Figure 4.3. Seamless Mobility and Roaming Protocols Unit Test Result

Figure 4.3 presents the unit test result for *Seamless Mobility* and *Roaming* protocols. Similar to other protocols, there is a transitive period at the beginning of the simulations; however it reaches steady state value of 3.28 seconds. Because of the public key operations, the delay value is higher than usual in these protocols. However, a client does not stop receiving service from the old access point until *Seamless Mobility* or *Seamless Roaming* protocols is finished. Therefore, a user does not experience a noticeable delay caused by a handover.

4.2.4. Unit Test Result for Packet Transfer

Packet Transfer is the mostly used protocol in the system. It is crucial to have small amount of network delay for this protocol because of often use. Unit test scenario of *Packet Transfer* protocol is that a client sends 10 packets of 512-byte every minute.

Figure 4.4 shows the unit test result for Packet Transfer protocol.

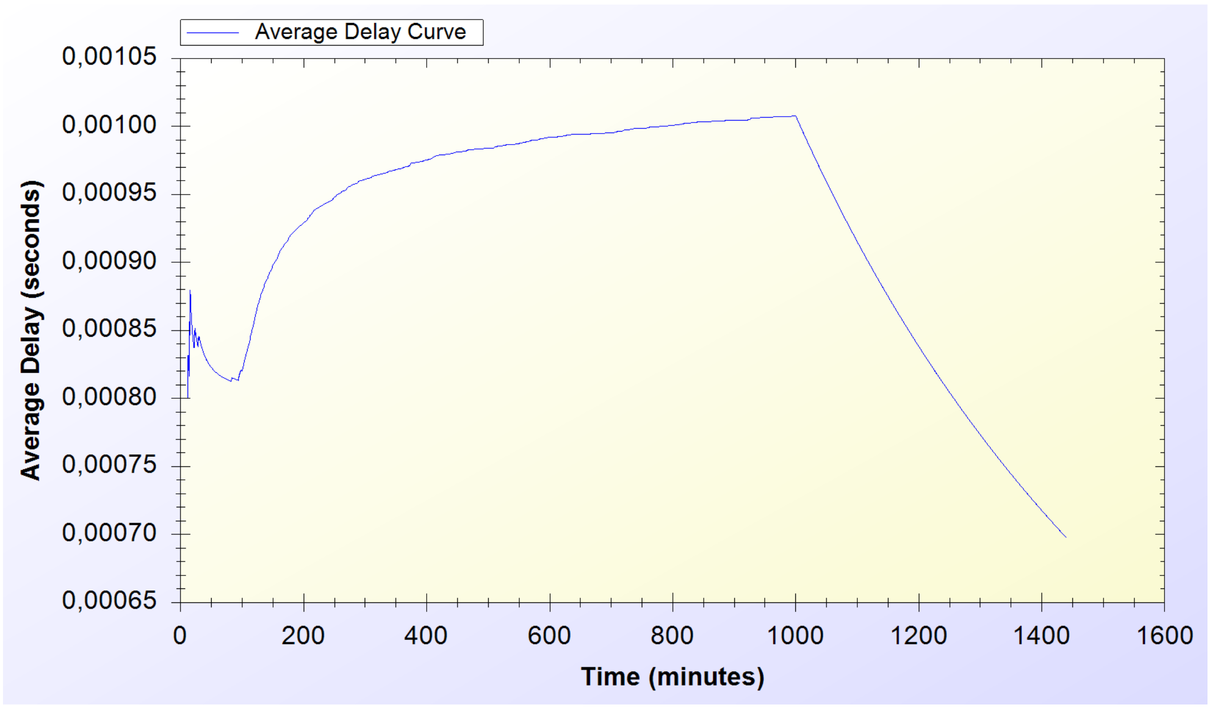

Figure 4.4. Packet Transfer Protocol Unit Test Result

Unit test gave a higher average delay value at the early parts of the simulation but expectedly it reaches a balance through time and shows a significant decrease by the end. Packets are received in a very short amount of time, which is around 0.0007 second.

4.2.5. Unit Test Result for Update Packets

Update Packets protocol takes place between AP and TTP. In this simulation access point updates the user info stored at operator. Figure 4.5 shows the average delay of *Update Packets* protocol over time.

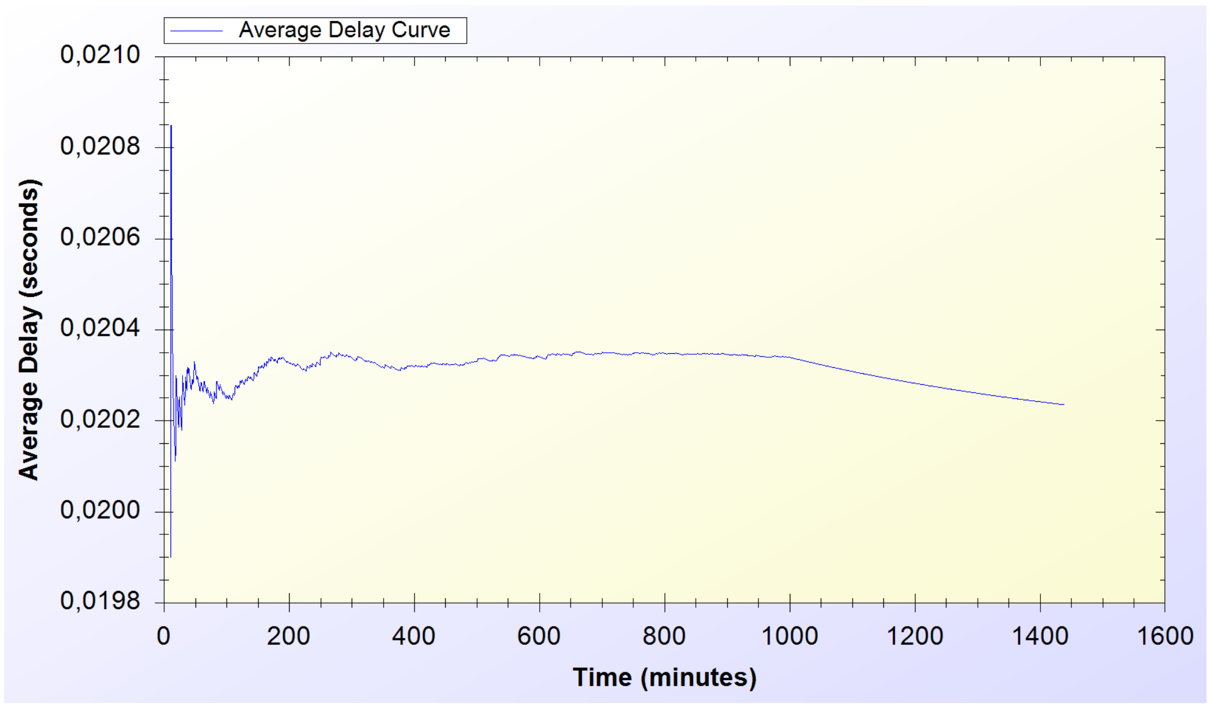

Figure 4.5. Update Packets Protocol Unit Test Result

In the simulation scenario, APs update operator once in every second. Our simulation showed that there is approximately 0.02 second maximum network delay for updating operator for the client usage.

4.3. USER MODELING AND MOBILITY IN REAL-LIFE SCENARIO

The proposed system intends to serve a variety of users (a.k.a. network clients). Network clients differ in their network usage frequency with respect to time of day, their mobility patterns and frequency of usage.

Certain kinds of actions are defined, such as authorization (initial or reuse of a connection card), disconnection, packet transfer (network usage), payment related roaming and payment related AP handover. All of these actions are triggered as a result of a random event. Connection and network usage related actions are triggered according to a two-state Markov Chain model. Roaming and handoff related actions are triggered by user mobility.

4.3.1. User Actions

In real-life scenario simulations, network usage related actions are modeled using two-state Markov Chain as shown in Figure 4.6. There are two states that a user could be in: *Connected* and *Not Connected*. State transitions or staying in the same state triggers some actions as described below.

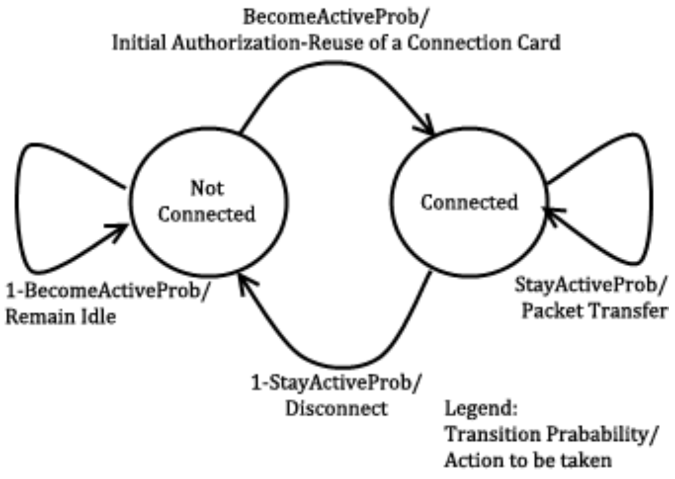

Figure 4.6. State Diagram of Clients

The initial state is *Not Connected*. In this state, the user switches to *Connected* state with the probability value of $BecomeActiveProb$. This state transition triggers *Initial Authorization* (if the CC is used for the first time) or *Reuse of a Connection Card* protocol (if the connection has been used before). In this way, the user starts consuming the network and getting the service. While in *Not Connected* state, the user stays in the same state with probability value of $1 - BecomeActiveProb$.

While in *Connected* state, the user remains connected (i.e. stay in the same state) with the probability of $StayActiveProb$. Staying connected triggers *Packet Transfer* protocol. In other words, the user continues to get service via the currently connected AP. In *Connected* state, transition to *Not Connected* state occurs with probability of $1 - StayActiveProb$. This transition disconnects the user via *Disconnection* protocol.

In this 2-state Markov chain model, the average connection duration, T_{con}, is calculated as the expected value of staying in *Connected* state, as given below.

$$T_{con} = \sum_{i=1}^{\infty}(1 - P_{SA}) \cdot i \cdot {P_{SA}}^{i-1} = (1 - P_{SA}) \sum_{i=1}^{\infty} i \cdot {P_{SA}}^{i-1} = \frac{1}{1-P_{SA}} \qquad (1)$$

Where, P_{SA} denotes $StayActiveProb$.

The expected value of staying in *Not Connected* state is the average idle time for a user between two connections. This value, T_{idle}, is calculated as follows.

$$T_{idle} = \sum_{i=1}^{\infty} P_{BA} \cdot i \cdot (1 - P_{BA})^{i-1} = P_{BA} \sum_{i=1}^{\infty} i \cdot (1 - P_{BA})^{i-1} = \frac{1}{1-(1-P_{BA})} = \frac{1}{P_{BA}} \quad (2)$$

Where, P_{BA} denotes $BecomeActiveProb$.

4.3.2. Client Types

Three different user types are outlined with different networking and mobility requirements. Considering whether they are working, studying or domestic provides the differentiation among user types.

The network usage within one day has been modeled in three time slots: (i) morning (06:00 – 13:59), (ii) daytime (14:00 – 21:59), and (iii) evening (22:00 – 05:59).

User types are described as follows:

- **Students:** This kind of clients uses network services mostly in the evening when they return back from school. Their possibility to use network services during morning and evening is relatively small comparing to mid-day time. Thus, the probabilities for being active are higher for evening. Students are assumed to be mobile at the beginning and end of the *daytime* slot since they go to their school. Until the end of the *evening* slot, students would more likely to get service in their homes in an immobile way.
- **Employees:** This kind of clients has routine lives. They are immobile and not so active during evenings. However, during the daytime, they are very active and use network services at their work places. Moreover, they are mobile as they commute to/from work from/to home at the beginning and end of the working times.
- **Domestics:** This type of users does not work outside and spend their time at home. Usually the domestics get Internet service in an immobile way. These users are highly active at all times.

The parameters of $StayActiveProb$ and $BecomeActiveProb$ are determined based on the abovementioned discussion about the client type characteristics and the time slots. These values are given below. The triplet $\{x,\ y,\ z\}$ specify the probability values for morning, daytime and evening, respectively.

$$becomeActiveProb < Domestic > \ = \ \{0.40,\ 0.60,\ 0.60\};$$

$$becomeActiveProb < Student > \ = \ \{0.20,\ 0.20,\ 0.80\};$$

$$becomeActiveProb < Employee > \ = \ \{0.20,\ 0.99,\ 0.20\};$$

$$stayActiveProb < Domestic > \ = \ \{0.90,\ 0.98,\ 0.80\};$$

$$stayActiveProb < Student > = \ \{0.30,\ 0.20,\ 0.98\};$$

$$stayActiveProb < Employee > \ = \ \{0.30,\ 0.99,\ 0.20\};$$

These values also determine the average connection duration and idle time by using Eq. 1 and 2. For example, a domestic client remains idle during daytime for $\frac{1}{1-(1-0.6)} = \frac{1}{0.6} = 1.67$ minutes between connections. Once connected, average connection time for this category is $\frac{1}{1-0.98} = \frac{1}{0.02} = 50$ minutes.

4.3.3. User Mobility and Timing

Real-time scenario covers Internet usage of 100-500 users in a 1-km^2 metropolitan area. The simulation time begins at 06:00 a.m. and lasts for 24 hours. Simulation time is divided into 3 parts considering morning, daytime and evening. Every part of the day has different statistical values for client behaviors.

Simulations are run for 1440 seconds, however every second in the simulation stands for 1 minute in real life.

In real-life scenario simulations clients are able to move from one location to another. The time and direction of their movement is selected at random but probabilities are affected by user roles. For example, when school is over, a student is most likely to move towards her target destination (e.g. her home).

Clients are assigned a random target access point. Every one of 100 access points has 3 initial clients. The client moves from its current access point to the target access point on the grid. An example movement pattern is shown in Figure 4.7. As a client moves from access point A to the access points B, if she needs to connect to the Internet, she forms up a new connection with the access point, which is the closest to the client's current location.

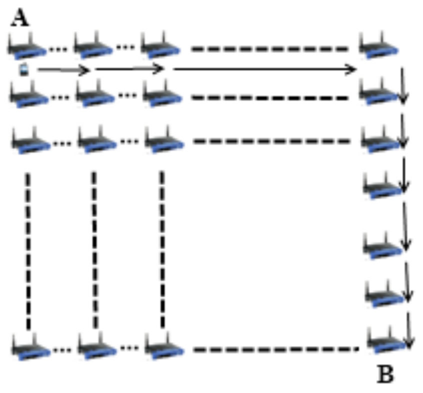

Figure 4.7. User Movement from A to B

In real-life scenario simulations, there are two operators and they have same amount of access points. In current simulations, each operator has 50 access points. The client executes handover or roaming if there is an active connection during movement between access points. In such a case, depending on the new access point's affiliated operator, user's movement triggers either *Seamless Mobility* or *Roaming* protocols. If new access point's affiliated operator is same as the one that client currently uses, and then it means the client would perform *Seamless Mobility* protocol for handover. Otherwise, the client would run *Seamless Roaming* protocol.

Clients are assigned uniformly distributed random speeds between 2 km/h to 6 km/h. The clients are assumed to move without a motor vehicle.

4.4. REAL-LIFE SCENARIO SIMULATION RESULTS

Real-life scenario consists of 100, 300 and 500 mobile clients trying to get network service in a 1km^2 metropolitan area. This scenario simulates an ordinary day with 24 hours. In these simulations clients have mobility patterns as mentioned before. Client's network usage frequency is affected by their socio-economical status. In following sections we will give latency values for each protocol. Finally, we will explain the overall burden on the system caused by the system's protocol.

4.4.1. Simulation Results of Protocols

In this section, we will briefly explain each protocol's run time performance. The protocols are performed on a network with 635 network entities. We have simulations for 100, 300 and 500 clients, 100 access points, 32 gateways, 2 operator servers and a server of the TTP. Every protocol in real-life simulations are executed simultaneously; therefore, real-life scenario simulation results have bigger latency values than unit tests.

In real-life simulations client models and mobility schemes are considered. The client roles and systematic mobility gave us realistic simulation results. In the beginning, every access point starts with 3 initial clients but then these clients move randomly in the metropolitan area and the initial setting do not remain as it was in the beginning.

4.4.1.1. Real-Life Scenario Simulation Result for Initial Authorization and Reuse of a Connection Card Protocols

Initial Authorization and Reuse of a Connection Card (Reuse-CC) protocols are used to start service from the system. Initial Authorization protocol only occurs at the beginning of the simulations; however, *Reuse-CC* protocol takes place frequently in the system.

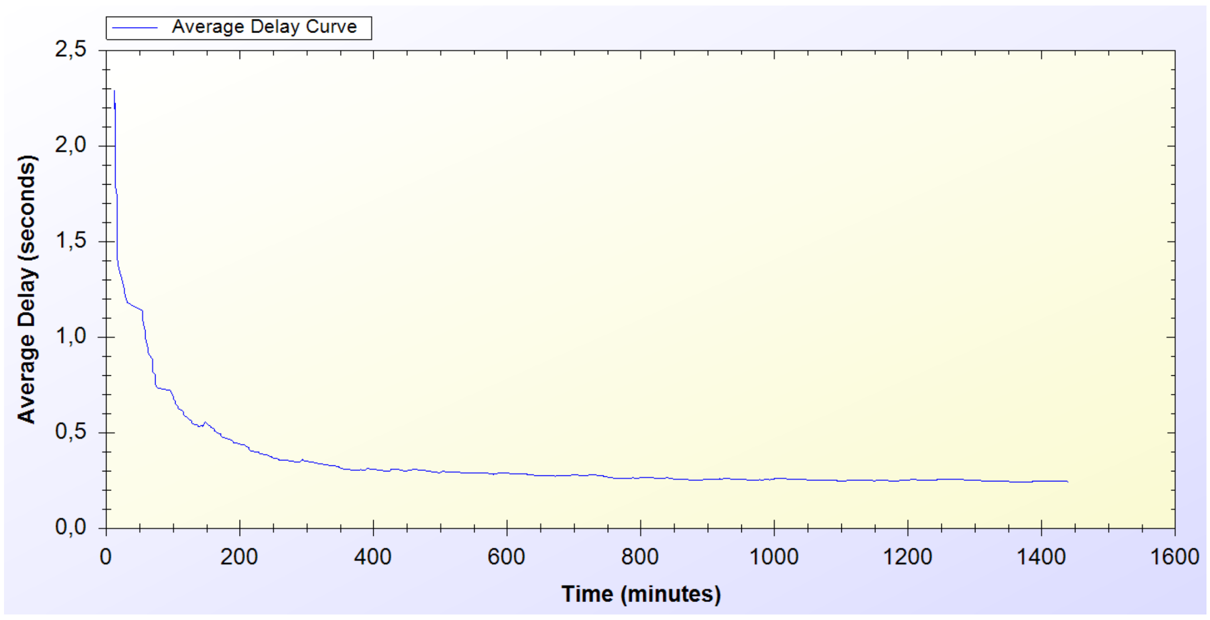

Figure 4.8. Real-Life Simulation Result for Initial Authorization and Reuse-CC Protocols (100 Clients)

Figure 4.8 shows Initial Authorization and Reuse-CC protocol performances in the real-life scenario simulation with 100 clients. The average delay for these protocols starts from 2.3 second. After some time, the protocol achieves a steady state performance of 0.3 second.

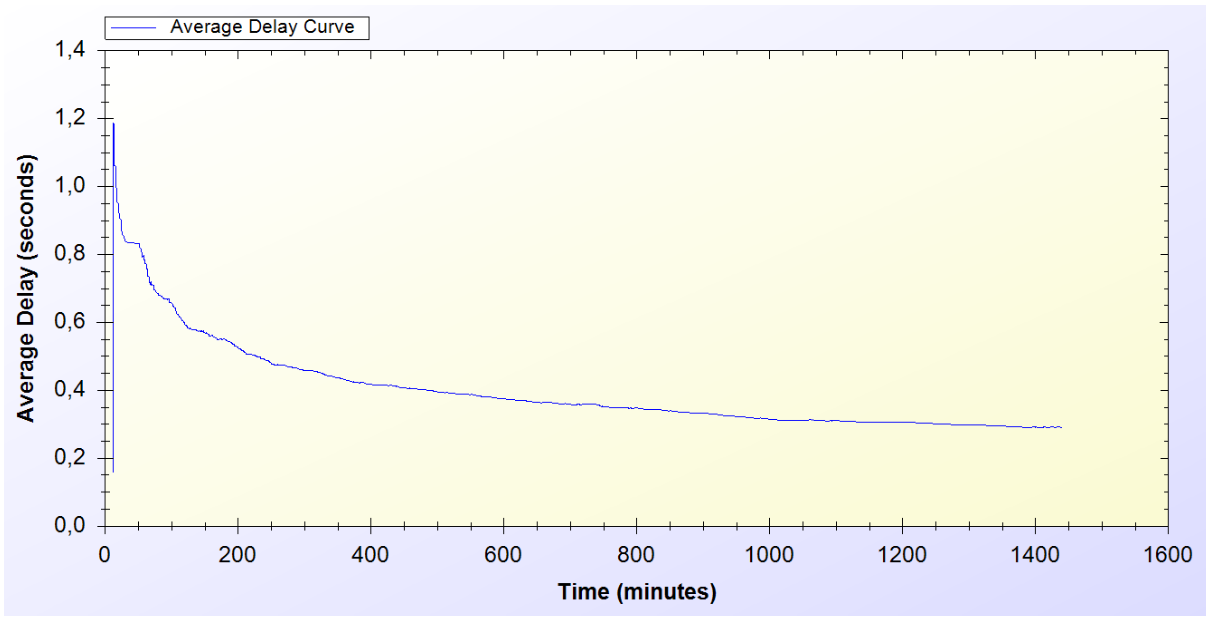

Figure 4.9. Real-Life Simulation Result for Initial Authorization and Reuse-CC Protocols (300 Clients)

Figure 4.9, shows the same kind of simulation results as the one previous. However, this time there are 300 clients in the system. With this setting, Initial Authorization and

Reuse-CC protocols show variation on delays between 0.2 and 1.2 seconds. By the end, average delay converges to approximately 0.35 second.

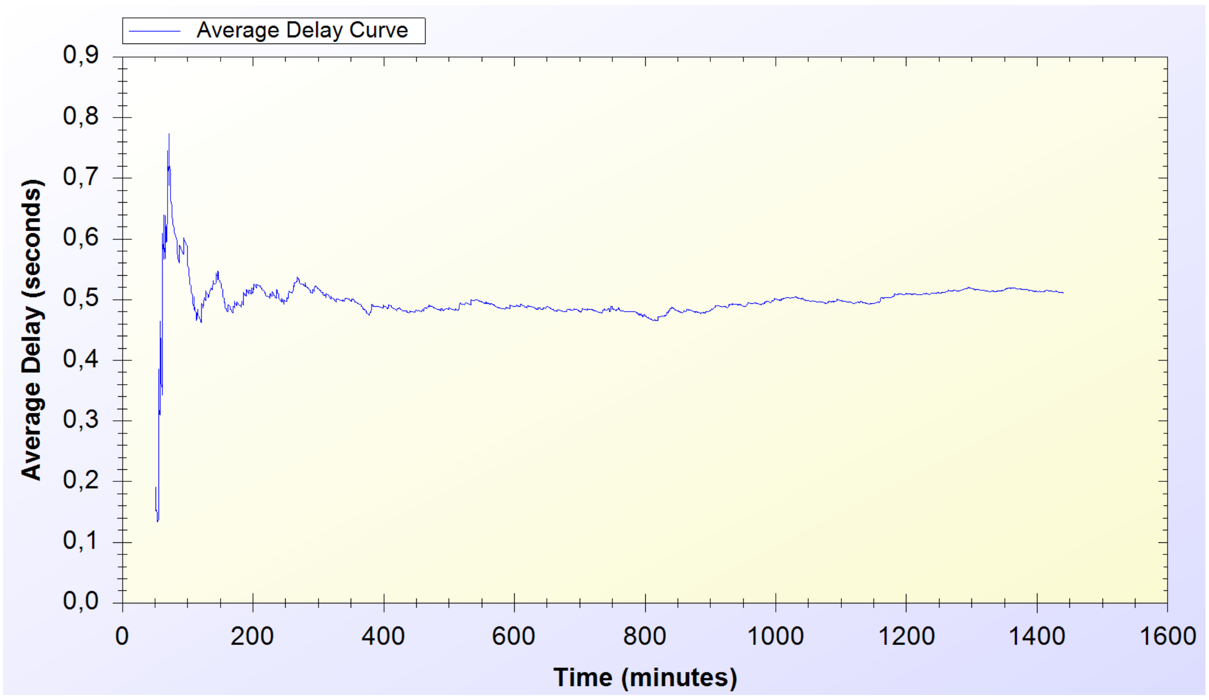

Figure 4.10. Real-Life Simulation Result for Initial Authorization and Reuse-CC Protocols (500 Clients)

Figure 4.10, shows Initial Authorization and Reuse-CC protocols' performances in a system with 500 clients. In this simulation the protocols show an initial disorder, average delay varies between 0.2 and 0.8 seconds but the steady state performance is achieved after 100 minutes around 0.5 second.

Performance comparison of these protocols is depicted on Figure 4.11. Comparing the results of the simulations with 100, 300 and 500 clients, we could see that network performance slightly decreases as the number of clients increase. However, the decline in system performance stays in reasonable level and shows an increase. The delay values start from 0.3 and gets up to 0.5 second.

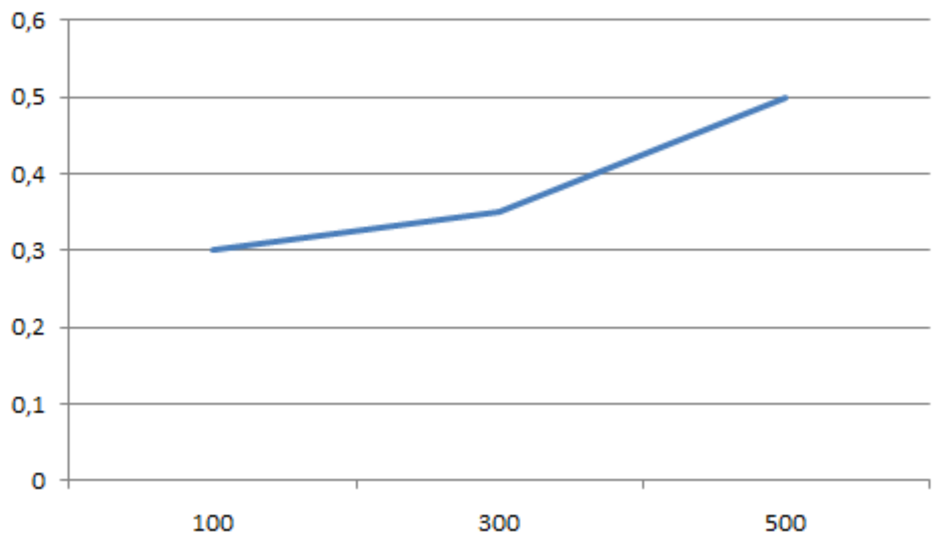

Figure 4.11. Initial Authorization and Reuse-CC Performance wrt. Number of Clients

4.4.1.2. Real-Life Scenario Simulation Result for Change Alias

Every active client uses *Change Alias* protocol in the system in every 60 minutes. The protocol is first used at 60th minute and it is used entire time of the simulation.

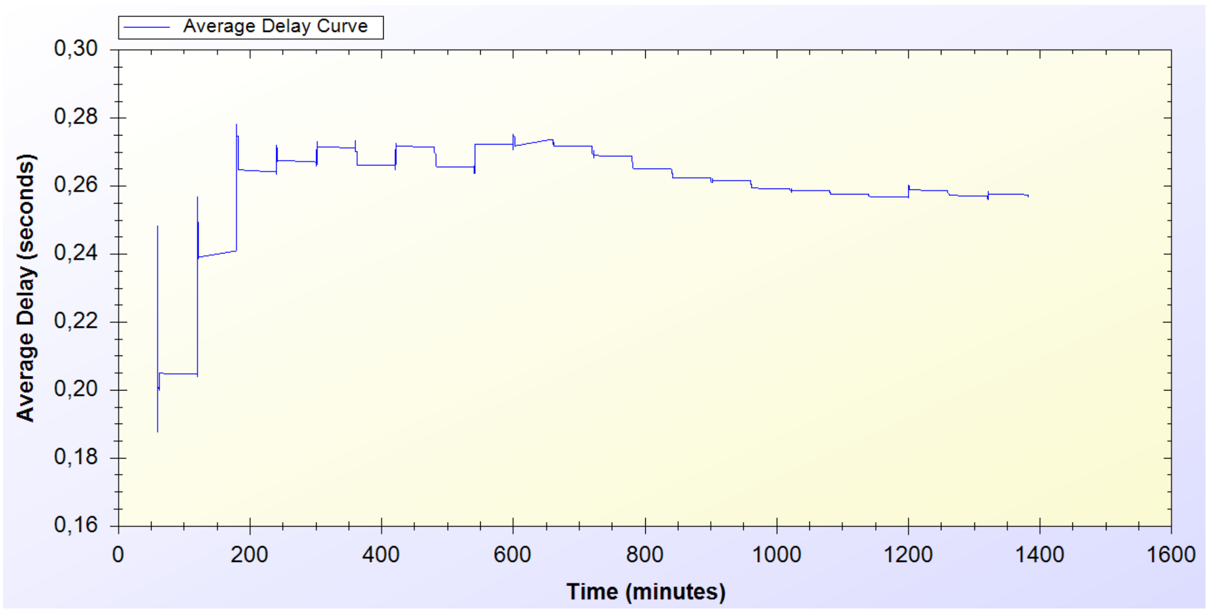

Figure 4.12. Real-Life Simulation Result for Changing Alias Protocol (100 Clients)

Figure 4.12 shows the *Change Alias* protocol performance in real-life scenario simulation with 100 clients. At the beginning of the protocol the delay for the protocol varies between 0.2 and 0.28 seconds. The average delay for the protocol converges to 0.26 second steady state.

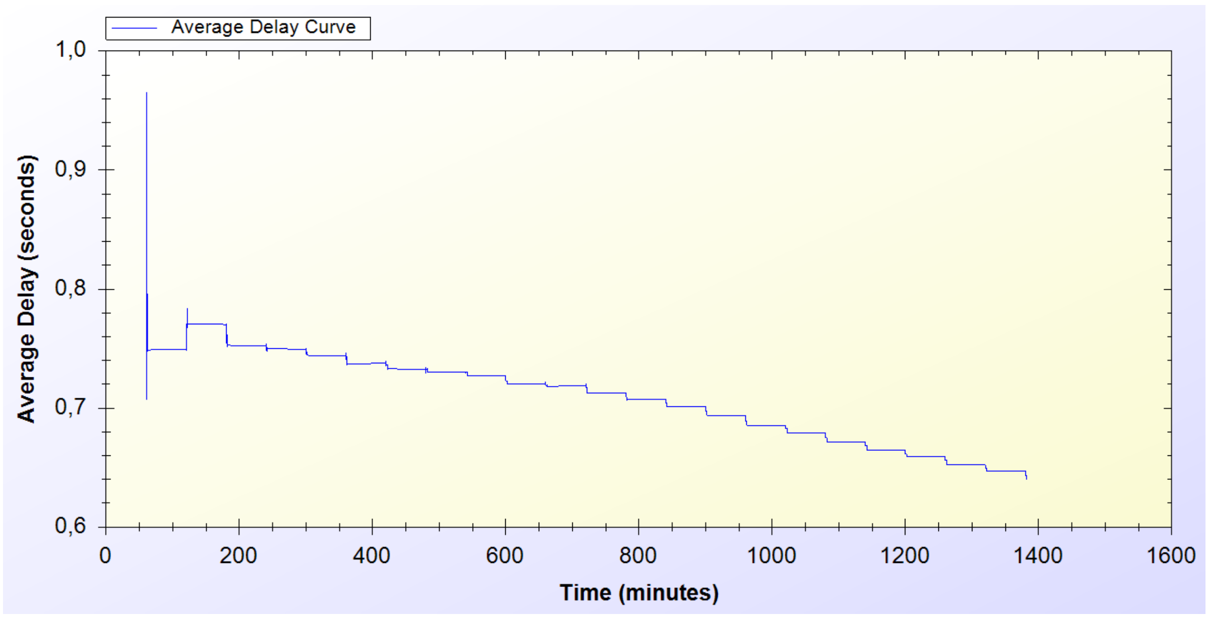

Figure 4.13. Real-Life Simulation Result for Changing Alias Protocol (300 Clients)

Figure 4.13, shows Change Alias protocol performance in real-life scenario simulation with 300 clients. Initial delay values vary between 0.7 and 1 second. The system achieves a steady state performance around 0.65 second after some disorder caused by initial deployment by the system entities. The Change Alias protocol delay values show a significant decline.

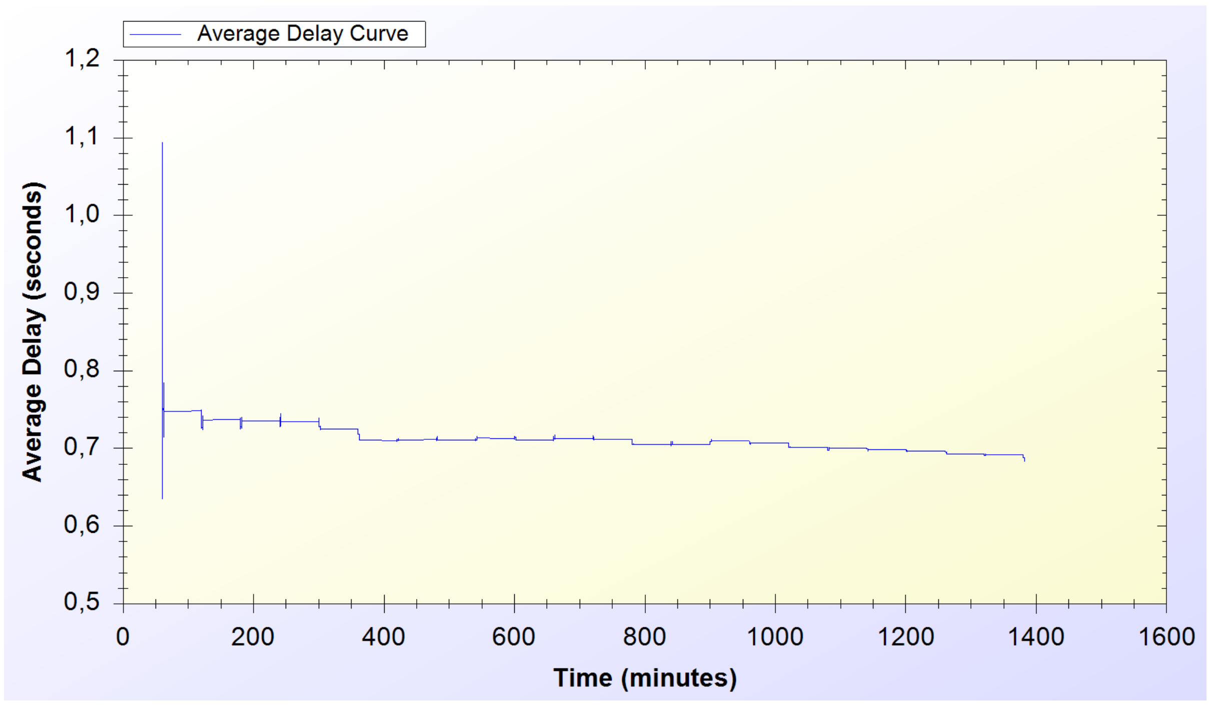

Figure 4.14. Real-Life Simulation Result for Changing Alias Protocol (500 Clients)

Figure 4.14, shows the *Change Alias* protocol performance in real-life scenario simulation with 500 clients. Initial delay values vary between 0.65 and 1.1 seconds. The protocols achieve a steady state delay around 0.7 second.

Change Alias protocol has 0.26 second of steady state performance for 100 clients; however, the steady state performance increases sub linearly and reaches up to 0.7 second of delay for 500 clients. The summary of Change Alias protocol performance comparison considering number of clients is depicted on Figure 4.15.

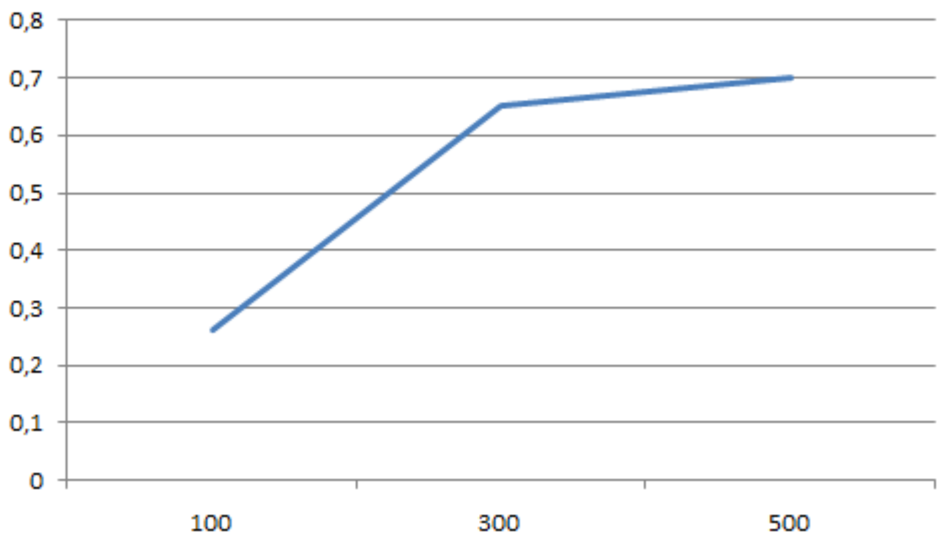

Figure 4.15. Change Alias Performance wrt. Number of Clients

4.4.1.3. Real-Life Scenario Simulation Result for Disconnection

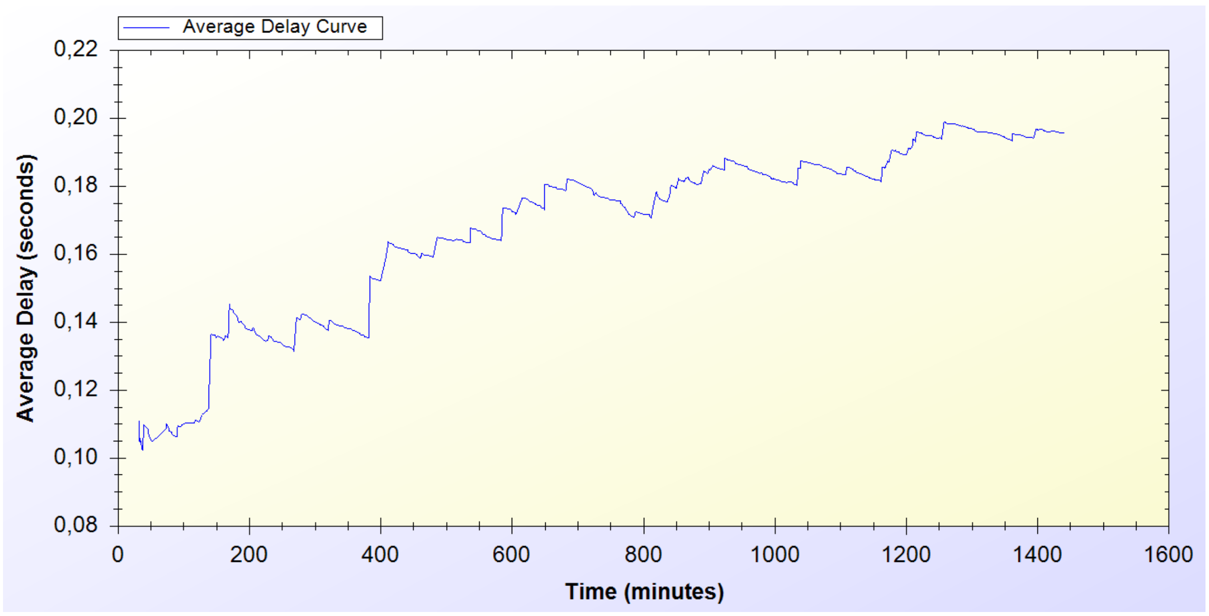

Figure 4.16. Real-Life Simulation Result for Disconnection Protocol (100 Clients)

Figure 4.16 shows Disconnection protocol performance in real-life scenario simulation with 100 clients. Disconnection protocol’s average delay starts from 0.1 second and shows a slight increase by the time passes. By the end the protocol reaches a final performance around 0.2 second.

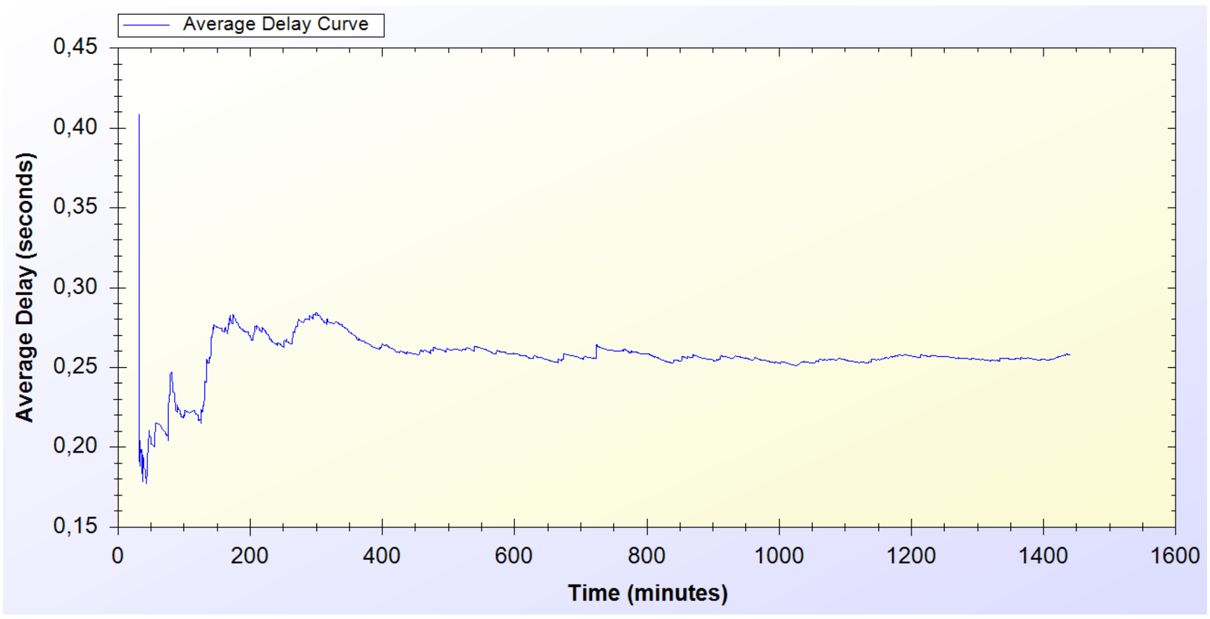

Figure 4.17. Real-Life Simulation Result for Disconnection Protocol (300 Clients)

Figure 4.17 shows Disconnection performance in a system with 300 clients. Initial delay values show variance between 0.2 and 0.4 seconds but then the average delay value converges to a value slightly over 0.25 second.

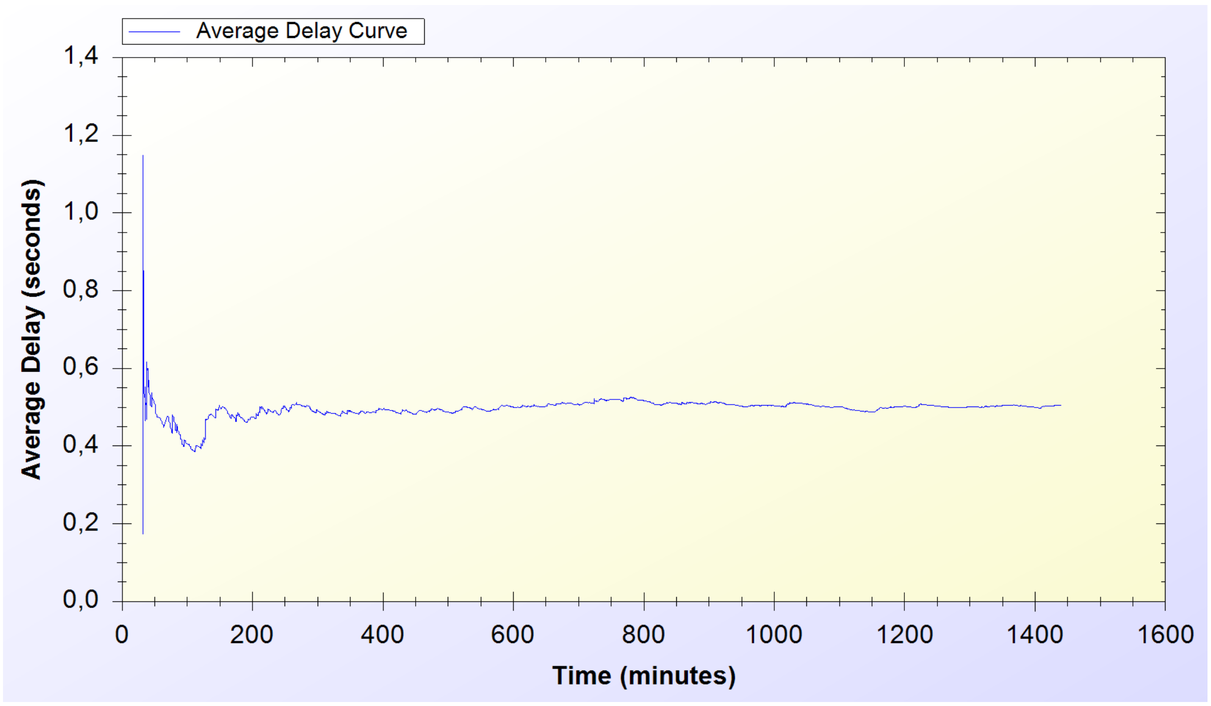

Figure 4.18. Real-Life Simulation Result for Disconnection Protocol (500 Clients)

Figure 4.18 shows Disconnection protocol's performance in a system with 500 clients. Initially, the protocol delay values differ between 0.2 and 1.2 seconds. Disconnection protocol achieves a steady state performance and causes network and computational delay around 0.5 second.

Disconnection protocol shows an increase from 0.2 second to 0.45 second. The protocol shows an increase as the number of clients gets higher. The comparison is shown in Figure 4.19

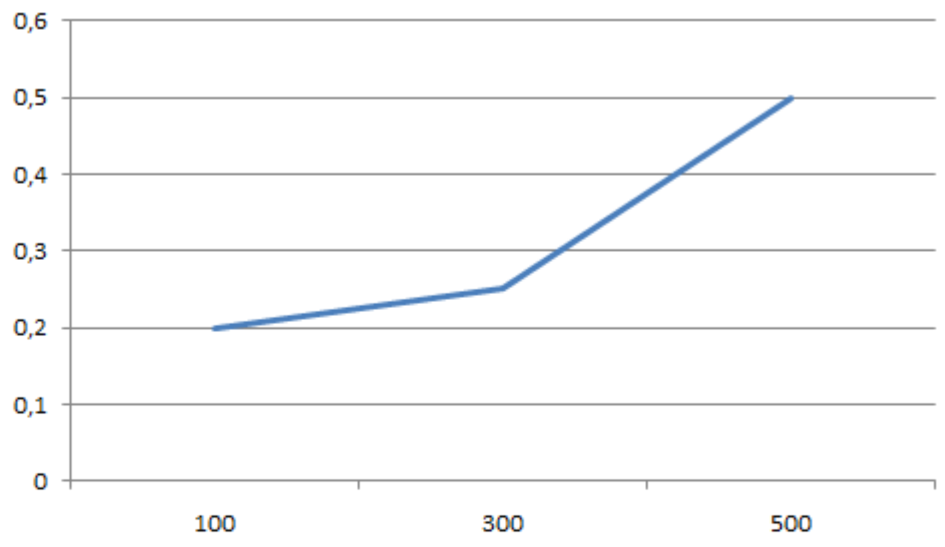

Figure 4.19. Disconnection Performance wrt. Number of Clients

4.4.1.4. Real-Life Scenario Simulation Result for Update Packets

Update Packets protocol is an end-to-end one-way protocol. It is expected to get lower delay values for this one. Only access points use *Update Packets* protocol and they send packets to TTP. The packets are sent every minute.

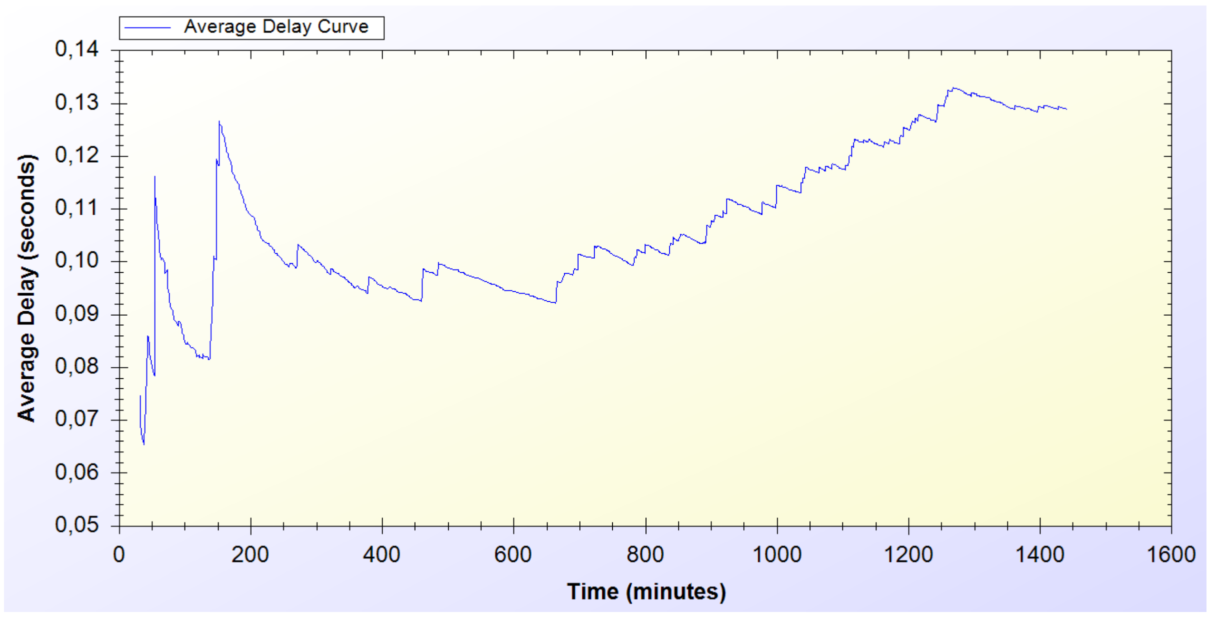

Figure 4.20. Real-Life Simulation Result for Update Packets Protocol (100 Clients)

As it is seen on Figure 4.20, which shows the Update Packets protocol's performance in a system with 100 clients, at the early stages of the protocol, the average delay value varies between 0.07 and 0.13 second but then after some time the protocol reaches a delay value around 0.13 second.

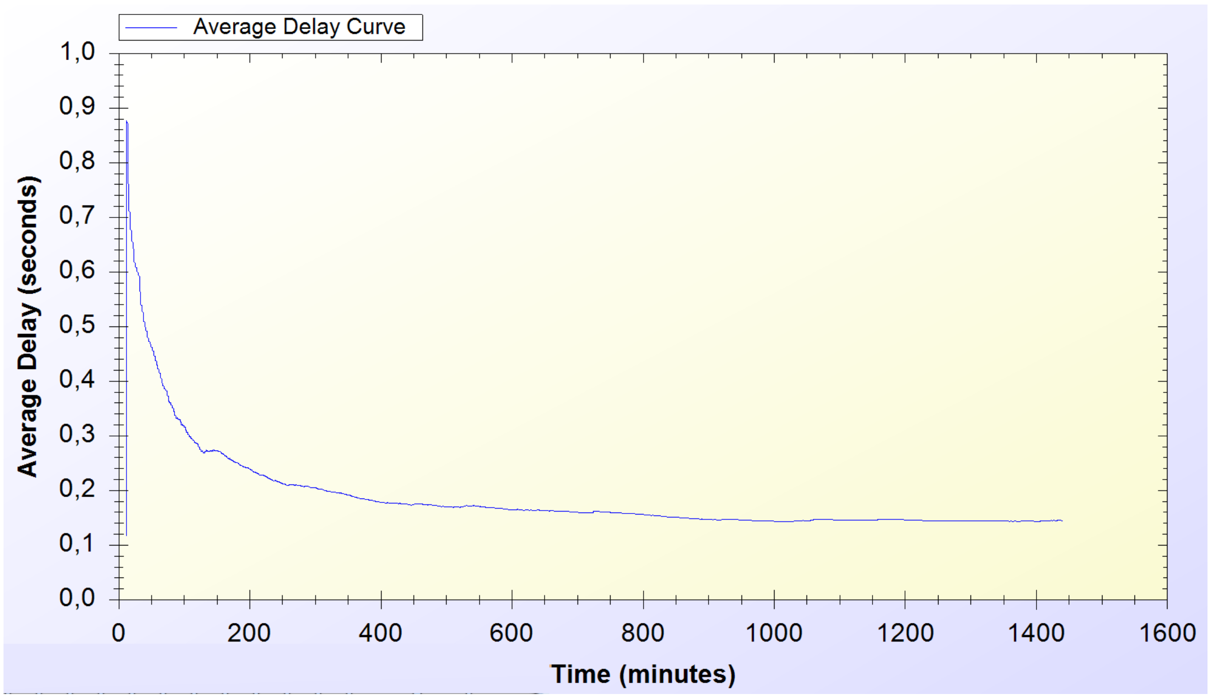

Figure 4.21. Real-Life Simulation Result for Update Packets Protocol (300 Clients)

Figure 4.21 shows Update Packets protocol performance in real-life scenario simulation with 300 clients. The system achieves a steady state performance by the end of the simulation and causes a delay value around 0.2 second.

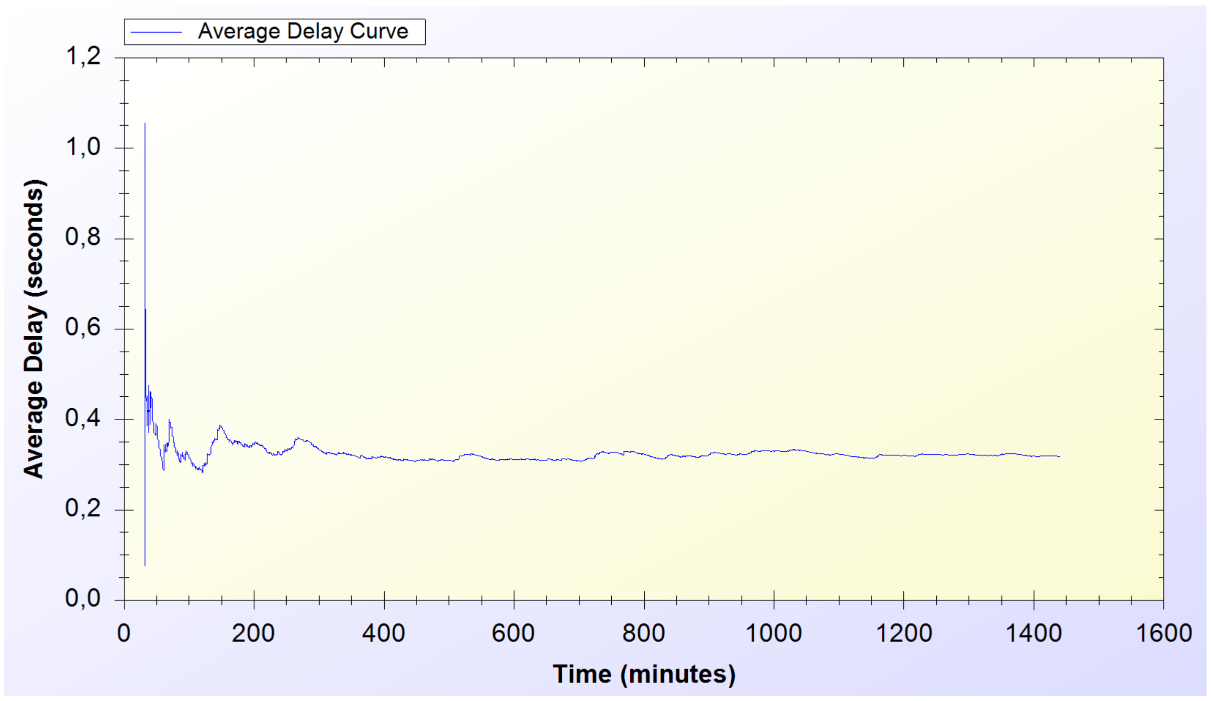

Figure 4.22. Real-Life Simulation Result for Update Packets Protocol (500 Clients)

Figure 4.22 shows *Update Packets* protocol performance under a system with 500 clients. Update Packets protocol achieves a steady state performance around 0.4 second with the previously mentioned settings.

Update Packets protocol shows an increase as the number of clients gets higher. The real-life scenario simulation for 100 clients results around 0.13 second of delay; whereas, the simulation for 500 clients gives around 0.35 second of delay. The increase occurs with tolerable values. Figure 4.23 depicts the comparison of Update Packets protocol performance considering the number of clients.

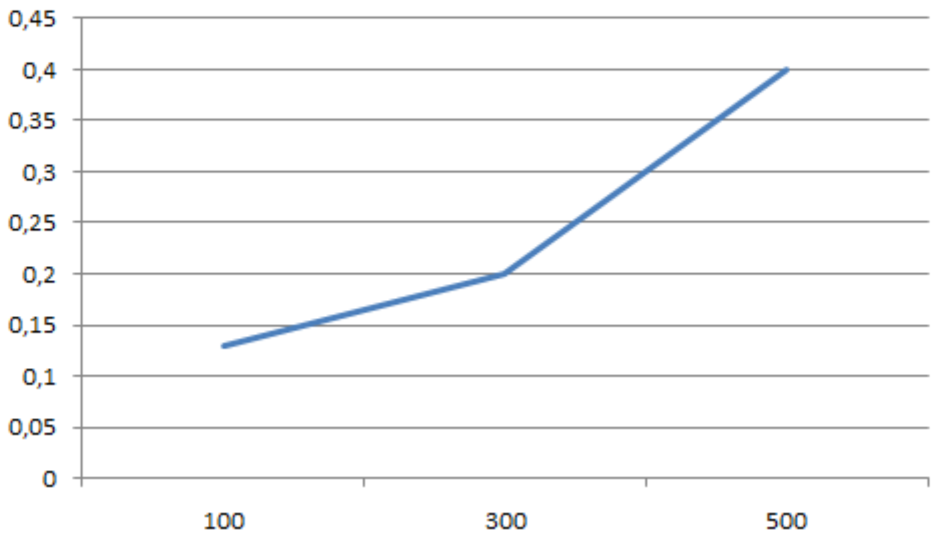

Figure 4.23. Update Packets Performance wrt. Number of Clients

4.4.1.5. Real-Life Scenario Simulation Result for Seamless Mobility and Seamless Roaming Protocols

Seamless Mobility protocol is used when a handover happens between access points. If these access points are belonging to the same operator then it means the client is using *Seamless Mobility* protocol.

Roaming protocol is used when a handover happens between access points. If these access points are belongings of different operators then it means the client is using *Roaming* protocol.

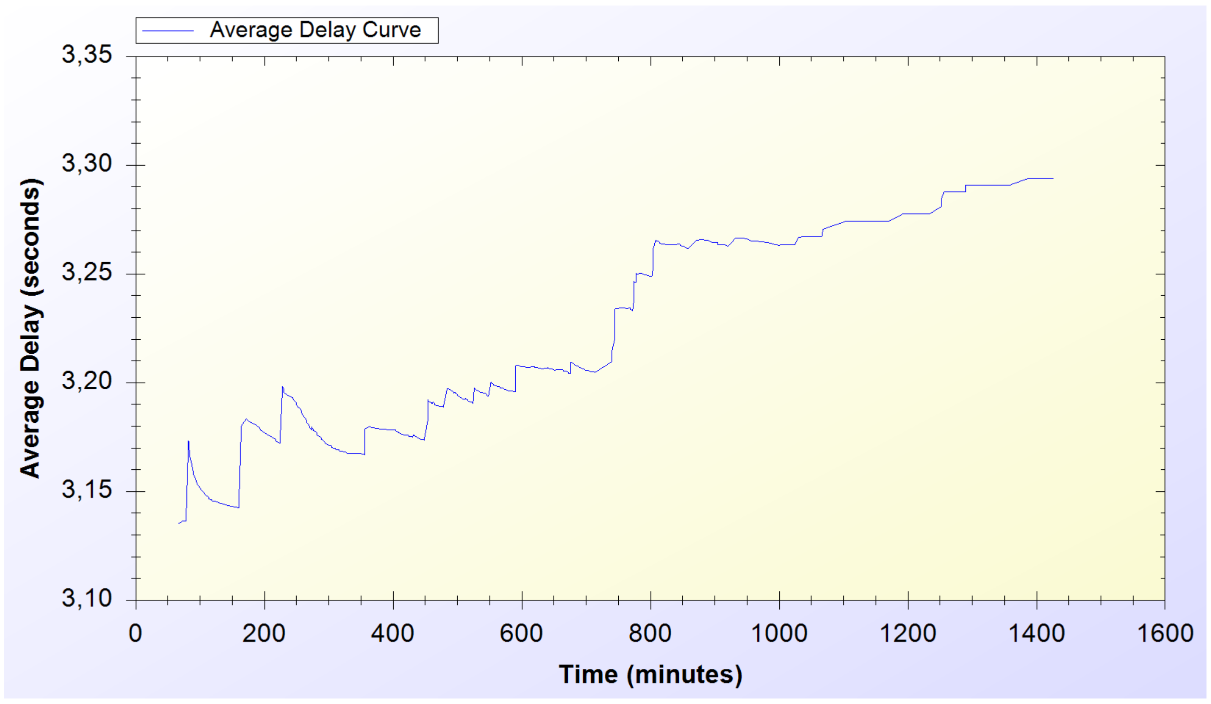

Figure 4.24. Real-Life Simulation Result for Seamless Mobility and Seamless Roaming Protocols (100 Clients)

Figure 4.24 shows Seamless Mobility and Seamless Roaming protocols performance on a system with 100 clients. It could be seen that these protocols has an initial average delay of 3.15 seconds and by the time passes the delay that caused by these protocols shows a slight increment. In steady state, a user spends around 3.30 seconds for the overall protocol run. Although this delay seems a bit high for a protocol that aims to be seamless, the protocol execution can overlap the actual usage with old access point. Thus, the client does not stop getting service from the old access point until she finishes all the handover process with the new access point.

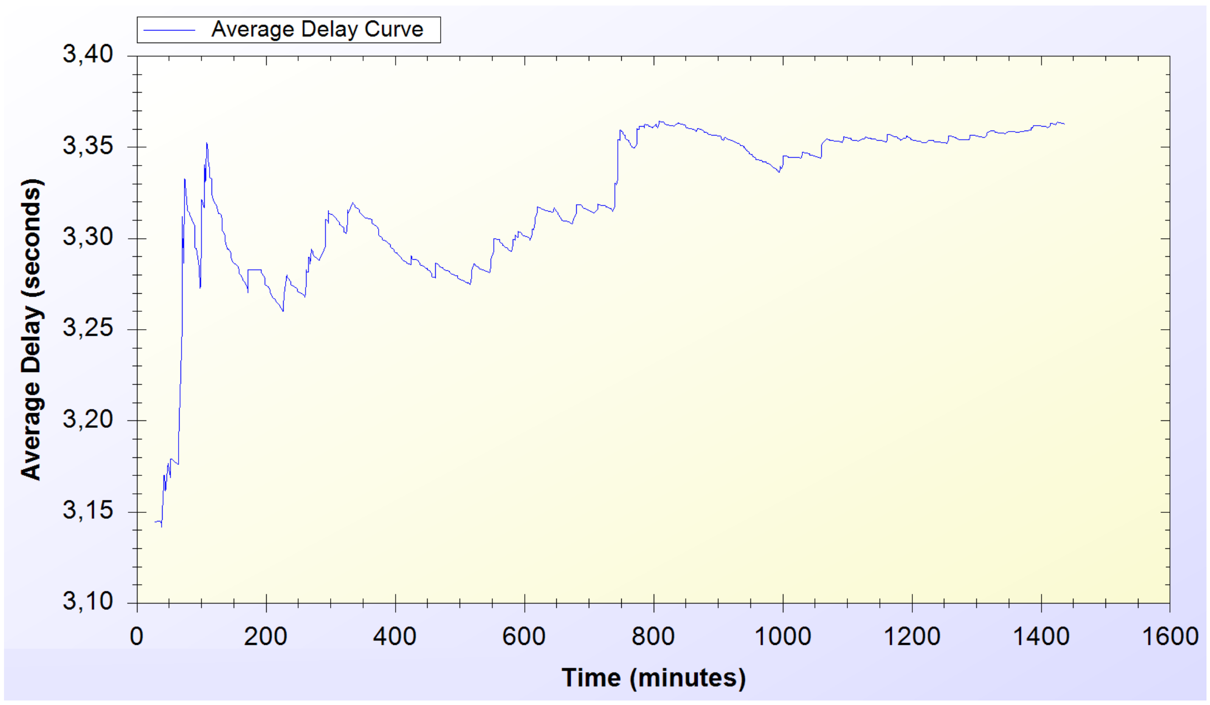

Figure 4.25. Real-Life Simulation Result for Seamless Mobility and Seamless Roaming Protocols (300 Clients)

Figure 4.25 shows the protocols performance on a network with 300 clients. Unlike the ones in previous simulation, the protocols reach a steady state performance around 3.35 seconds of average delay.

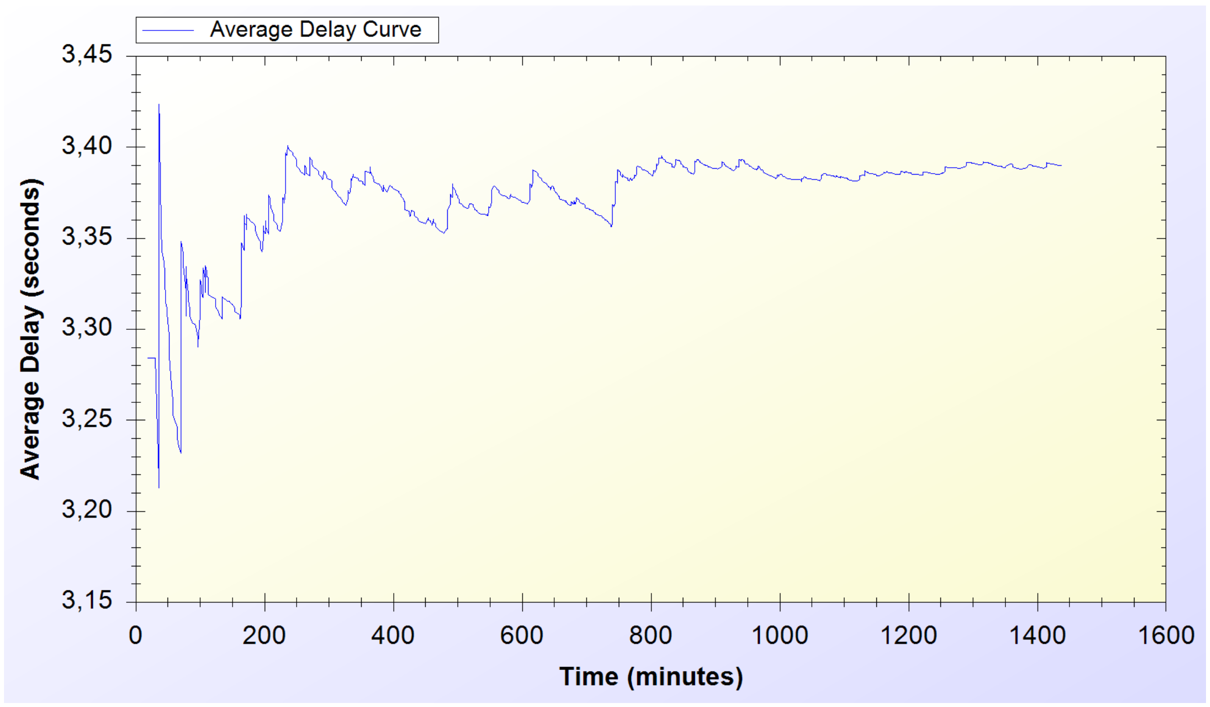

Figure 4.26. Real-Life Simulation Result for Seamless Mobility and Seamless Roaming Protocols (500 Clients)

Figure 4.26 shows Seamless Mobility and Seamless Roaming delays in a system with 500 clients. In this system, the protocols achieve a steady state performance at 3.40 seconds.

Different real-life scenario simulations for 100, 300 and 500 clients show that the delay values for *Seamless Mobility* and *Seamless Roaming* protocols start from 3.3 seconds and reach 3.4 seconds, the increase occurs linearly. These protocols do not show a significant change as the number of clients gets higher. Figure 4.27 depicts the change in the performance of the Seamless Mobility and Roaming protocols performance as the number of clients change.

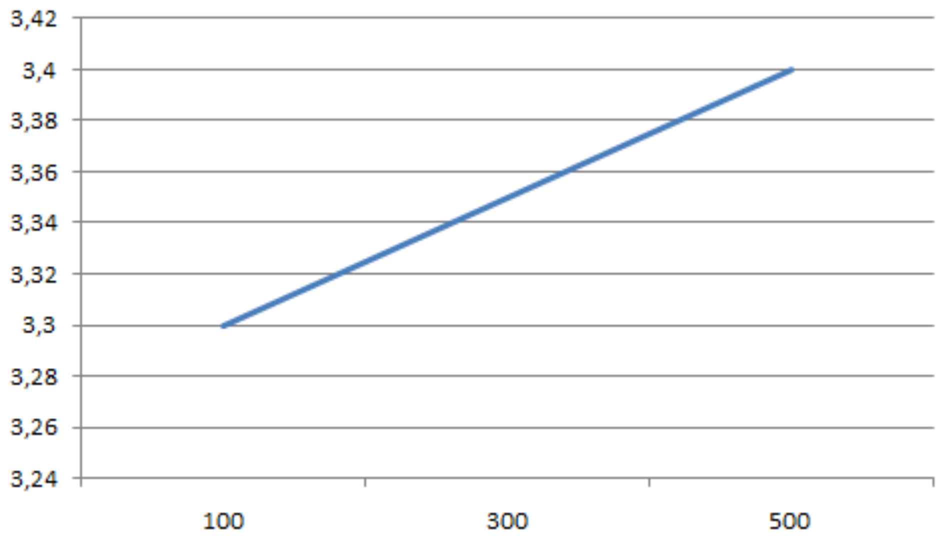

Figure 4.27. Seamless Mobility and Seamless Roaming Performance wrt. Number of Clients

4.4.1.6. Real-Life Scenario Simulation Result for Packet Transfer

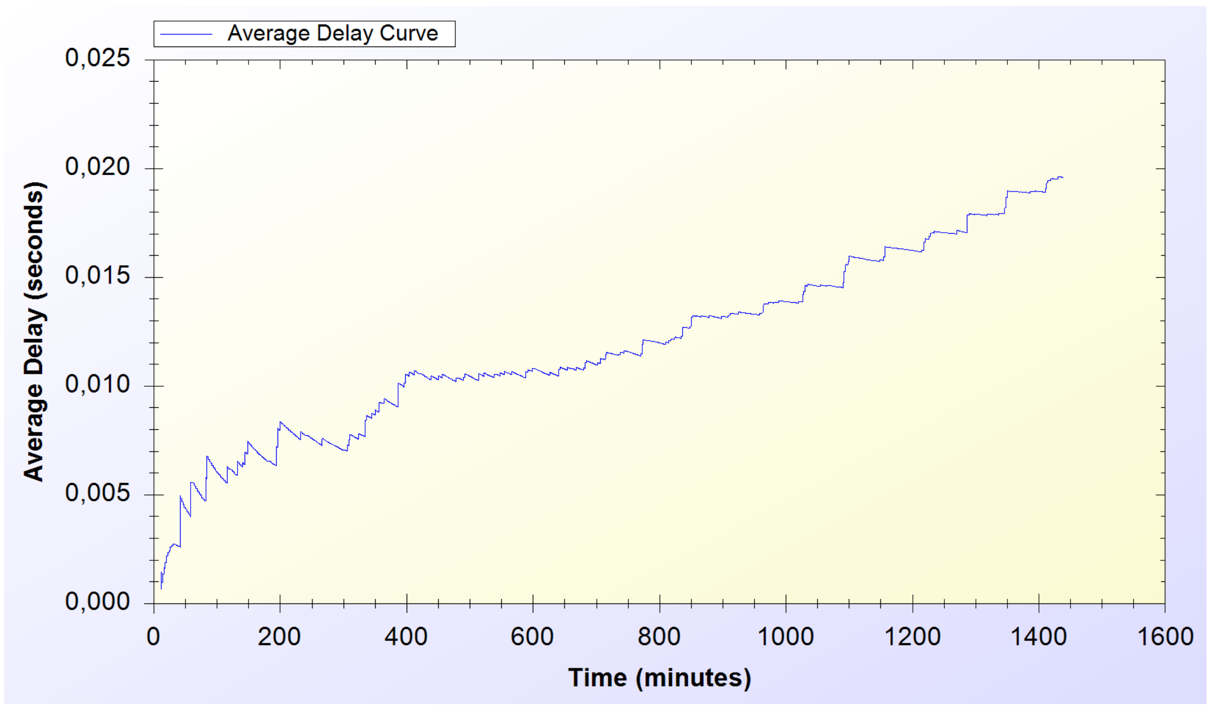

Figure 4.28. Real-Life Simulation Result for Packet Transfer Protocol (100 Clients)

Figure 4.28 shows Packet Transfer protocol performance in a network with 100 clients. The average delay value of the protocol starts from 0.002 second at the beginning of the simulation. The protocol delay linearly increases over time and causes 0.02 second of network delay at the end.

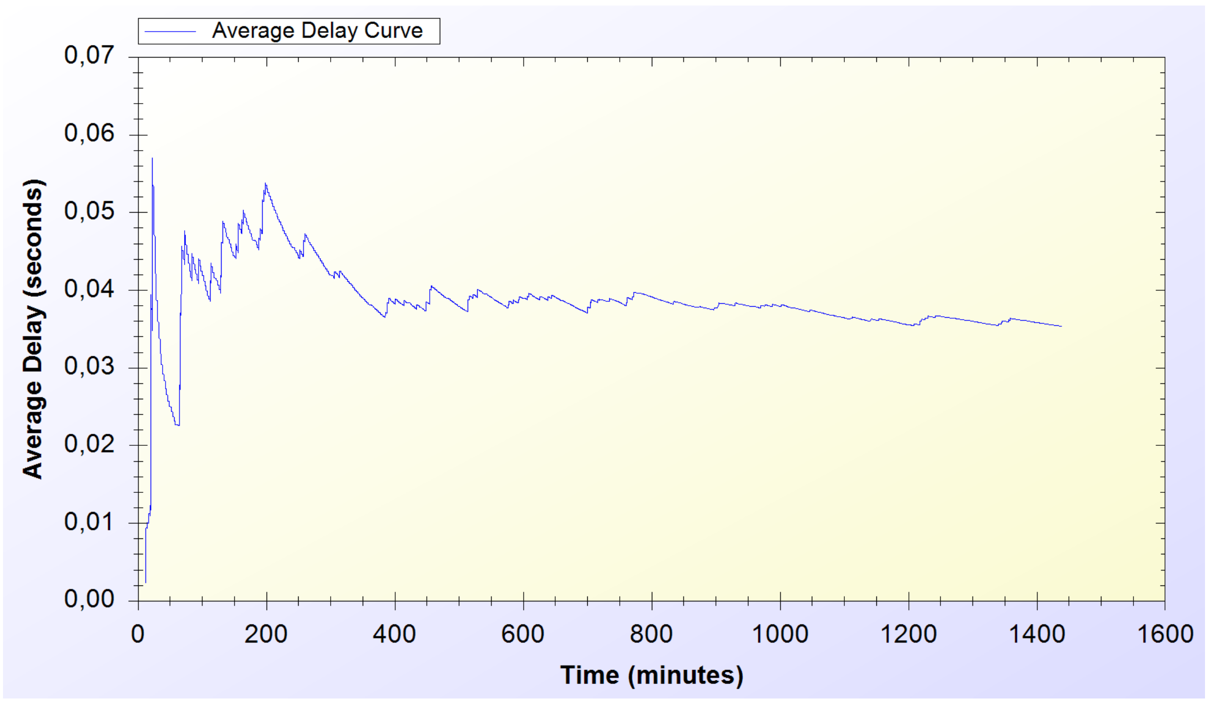

Figure 4.29. Real-Life Simulation Result for Packet Transfer Protocol (300 Clients)

Figure 4.29 shows Packet Transfer protocol performance in a network with 300 clients. Initial performance of the protocol is instable but it achieves steady state around 0.04 second of average delay.

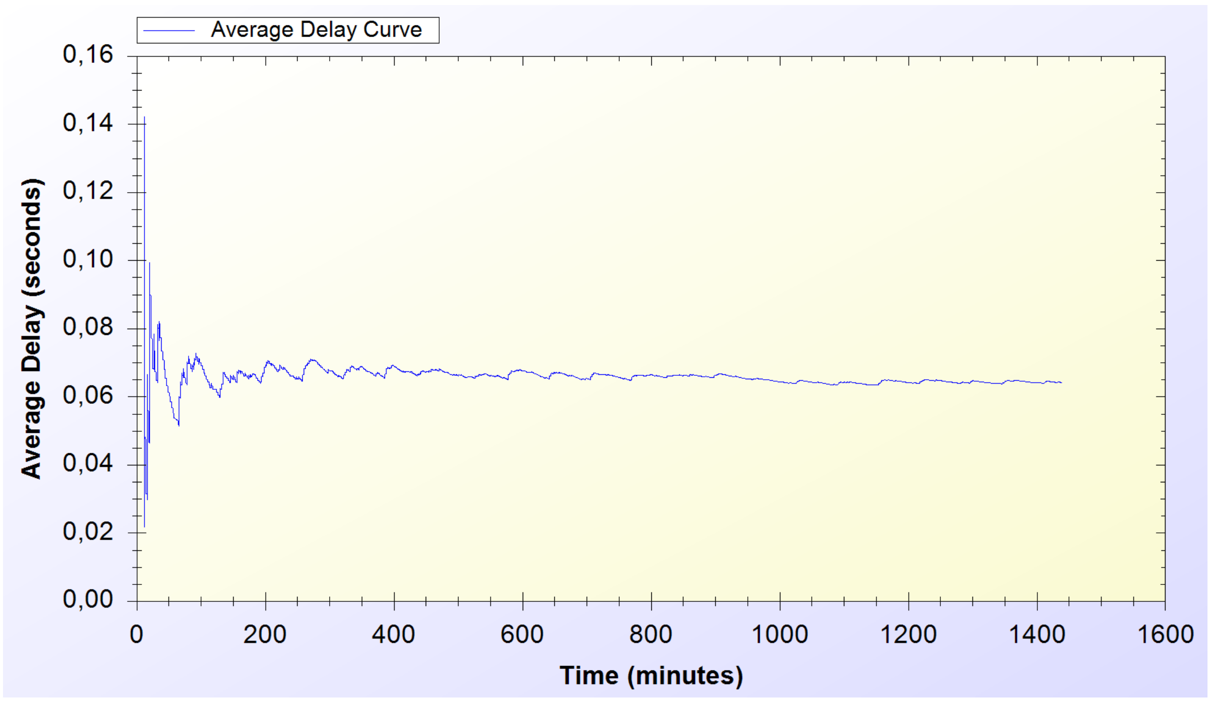

Figure 4.30. Real-Life Simulation Result for Packet Transfer Protocol (500 Clients)

Figure 4.30 depicts the performance of Packet Transfer protocol in a system with 500 clients. The protocol achieves a steady state average delay around 0.06 second.

Packet Transfer protocol delay linearly increases as the number of clients increase. The protocol delay is 0.02 for 100 clients; whereas, it is around 0.06 second for 500 clients. Figure 4.31 shows the change in Packet Transfer performance as the number of clients increase.

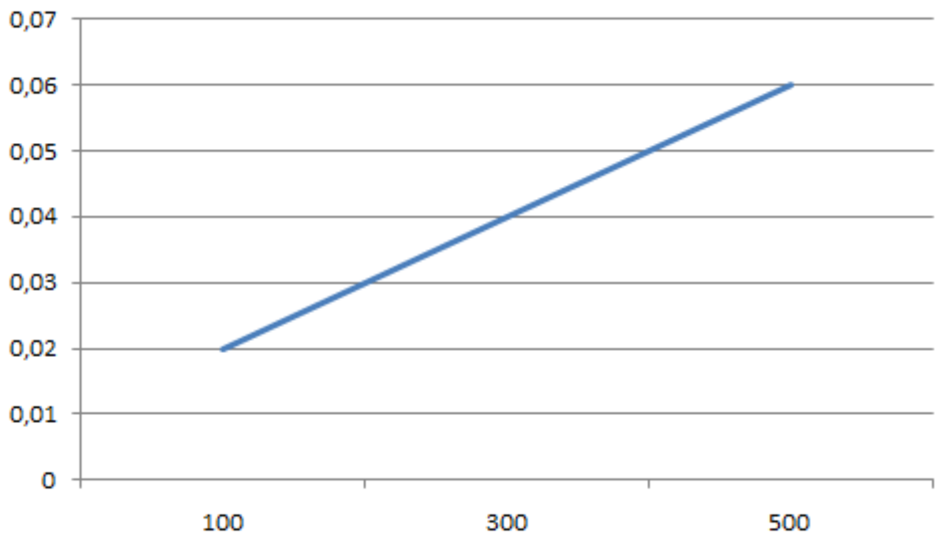

Figure 4.31. Packet Transfer Performance wrt. Number of Clients

4.4.2. Overall Burden of the System

Real-life scenario simulation for 500 clients provided the results in Table 4.6. Charts on Figure 4.32 and Figure 4.33 are drawn exploiting the results in Table 4.6 In average; over 1000 minutes of Internet service needs a delay of 12 to 15 minutes of waiting; which yields approximately 1.5% overhead.

Table 4.6: Simulation Results for Client Types

	Total Internet Usage Time	Total SSPayWMN Delay	Average Internet Usage Time for a Client	Average SSPayWMN Delay for a Client
Students	160858 Minutes	2660 Minutes	963 Minutes	15 Minutes
Employees	168296 Minutes	2062 Minutes	1013 Minutes	12 Minutes
Domestics	176792 Minutes	2558 Minutes	1058 Minutes	15 Minutes

Difference between client types affects the system usage of the clients. The probabilistic values, which are mentioned in Section 4.2, determine the system usage frequency of the clients. As it is seen on Table 4.6, domestics are the most active clients in the system. Employees and students follow domestics in that sense. On Table 4.6, the simulation results are grouped into two subgroups, which are: Total Internet Usage and Average Internet Usage. Total Internet Usage means the overall sum of network usage of these 500 clients in a day. When we analyze the delay values we see that a client experiences a delay, which is approximately 1.5% of the total amount of received service.

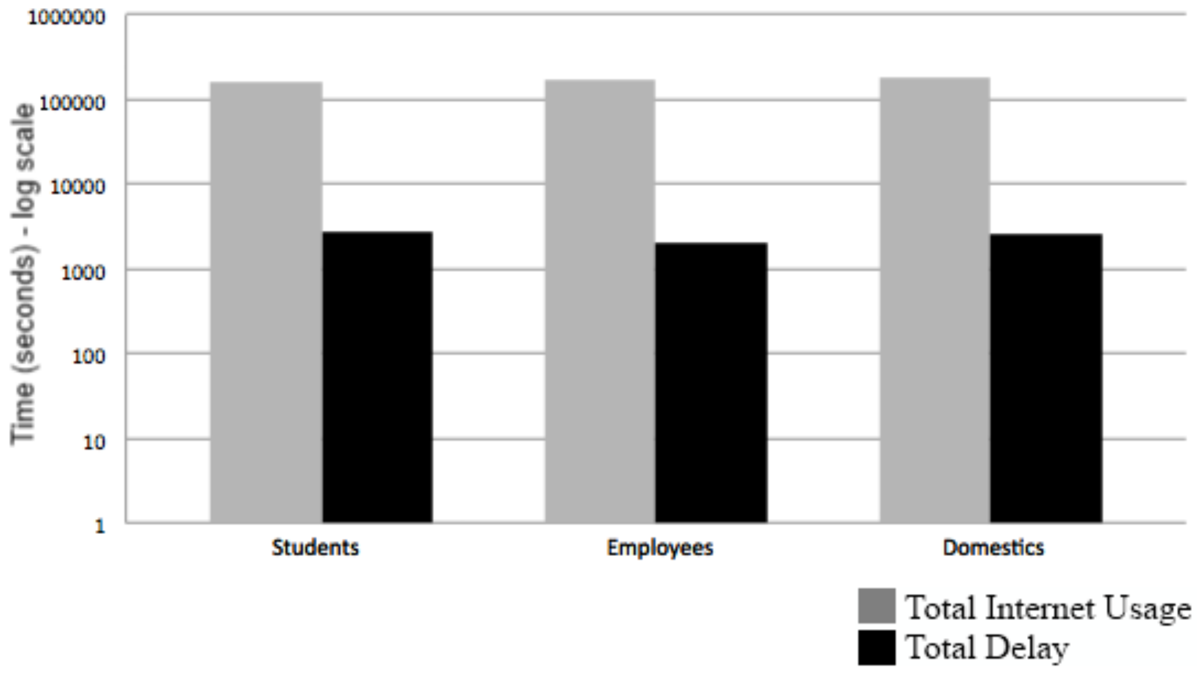

Figure 4.32. Total Amount of Service Usage Times for Client Types vs. Total Delays

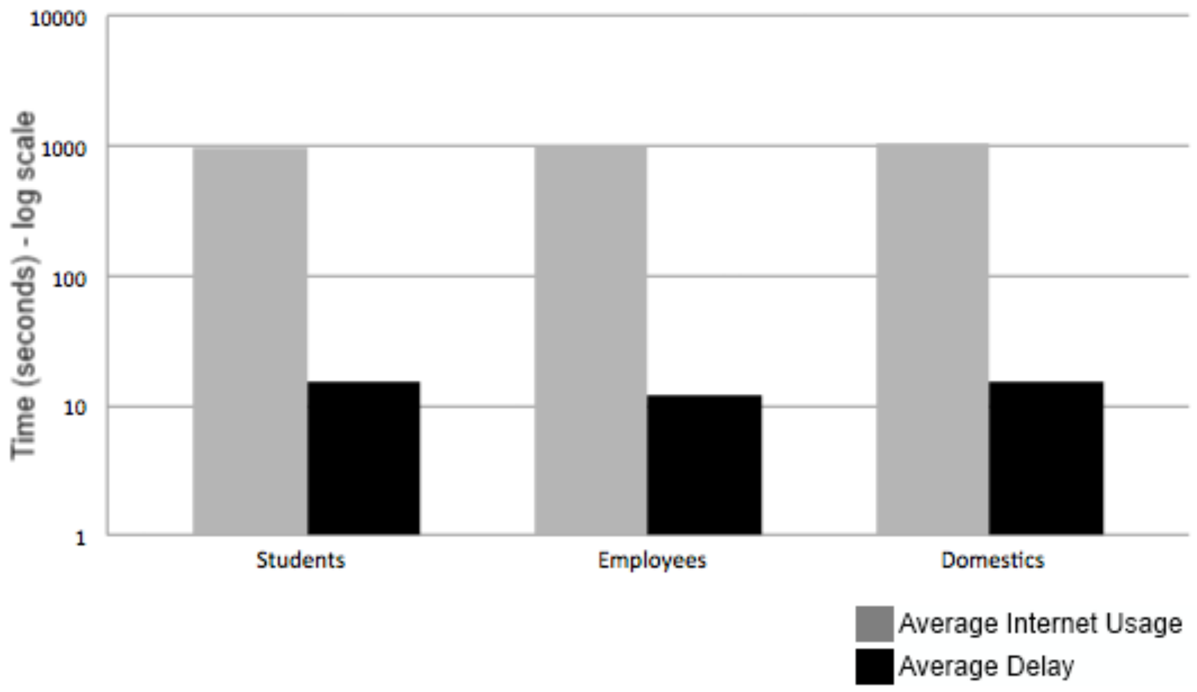

Figure 4.33. Average Service Usage Times for Client Types vs. Average Delays

As described before the clients are grouped into 3 subgroups. The client roles and probabilistic values affect their behavior in the system, which results difference between overall values of the simulations.

Figure 4.32 and Figure 4.33 shows the overall results for real-life scenario simulation. Figure 4.32 shows comparison of minutes clients used as idle or active. Figure 4.33 shows the average value for the clients of the same group.

5. DISCUSSIONS

In Section 2 the requirements for a secure and seamless pre-payment scheme were discussed. In this section the success of the proposed system on meeting the requirements and simulation results are discussed.

Wide Coverage

The main strong side of WMNs is economical wide network coverage. This objective was important since the proposed system intends to serve high amount of users within a large metropolitan area. To test SSPayWMN's success on providing network service within a wide area, we have used ns-3 simulator. In our simulations, all of the mesh routers cover a circular area of 100 meters of radius. In our simulations 1 km^2 area is covered. The size of the area could be widened with more access points.

We have covered a wide area with access points as well as we have granted Internet access for large amount of users. The real-life scenario simulations ensured that SSPayWMN is a scalable system for large amount of clients and remains stable when the number of clients increases.

Seamless Mobility and Roaming

Seamless Mobility and Roaming property of the system describes the ability of the users to handover between access points without experiencing a noticeable delay. To provide seamless mobility and roaming, we have implemented *Seamless Mobility* and *Seamless Roaming* protocols. These protocols transfer the current hash token of the client to another access point. The client does not stop getting service from the old access point until these protocols are finished. Whenever these protocols finish their process, client starts to get service from the new access point. Usage of these protocols prevents clients to wait for another authentication and disconnection processes.

Anonymity

This property of the system is required to keep the clients anonymous; therefore, this property prevents any adversary to compromise a client's real identity or any other sensitive information. To provide anonymity, SSPayWMN employs aliases. Only TTP and the client herself could relate the aliases to client's identity. We have computed the aliases using a hash function over an output of XOR operation on SN and a random Nonce. In this way SN remains a secret because of the irreversibility property of the hash functions. For law-enforcement reasons, users must give their identities to Trusted Third Party (TTP) for getting connection cards. Therefore, as far as TTP keeps clients' identities secret, users can remain anonymous.

The delay values for Seamless Mobility and Seamless Roaming protocols are higher than the other protocols; however, a client does not stop to receive service from the old access point until these protocols are finished. Therefore, there does not occur a noticeable network or computational delay for a client that affects her system usage.

Untraceability

As mentioned earlier, our system employs aliases to provide anonymity. However, in case of an adversary compromises the client's alias, the adversary could trace all the actions of that client. Untraceability is achieved in two steps: the first one is untraceability of client aliases; the second one is the untraceability of client's hash tokens. The first part of is achieved by forcing the clients to change their aliases periodically. Our way of alias generation prevents any relation between sessions of the clients. The second part of untraceability is achieved by usage of anonymized subhash chains. These hash chains have limited amount of hash tokens that are enough for a client until the next change alias round. Every time a client changes her alias she becomes unlinkable with previous sessions. SSPayWMN supports untraceability to some extent, which is a significant contribution for prepayment systems on WMNs.

Mutual Authentication

This property is required to provide security of the client and prevent an adversary to mimic an access point. Furthermore, this property is necessary to prevent an adversary to mimic a valid client. Mutual Authentication has two steps: the first one is the client's authentication and the second one is the network's authentication. The network authenticates clients in *Initial Authorization* and *Reuse-CC* protocols. Clients send their SN and corresponding hash tokens to get authenticated. In the system, the client and the TTP are the only parties who know these values. Therefore, when client provides these values, she authenticates herself. On the other hand, in *Access Point Authentication* protocol access point also proves that it knows the current hash token of the client. Since, hash functions are irreversible, no one but the owner of the hash chain could not know the current hash token by looking at the previous ones. *Access Point Authentication* protocol provides the second part of the mutual authentication property.

Two-way honesty

This property means that in SSPayWMN, it not possible for clients to con operators; whereas, it is not possible for operators to overcharge clients. In SSPayWMN, we addressed this objective by using hash tokens as e-currency. Because of the irreversibility property of hash algorithms it not feasible to find the next hash token by using the previous hash tokens. In case of a denial of a provided service by the client, operators could show the logs of the provided network service with the corresponding hash token. Therefore, it is not possible for a client to deny provided network service. On the other hand, operators could not overcharge their clients because an operator could not know the next hash token in the hash chain by exploiting the previous hash tokens.

Performance

The performance of SSPayWMN has been evaluated with simulations using ns-3. Two groups of simulations are performed: unit tests and real-life scenario simulations. Unit tests ensured the stable performance of the protocols in stand-alone run; whereas, real-life scenario simulations ensured the stable system performance in real-life situations. We have conducted real-life scenario simulations for different number of clients: 100, 300 and 500. We have compared the protocol performances considering the change in clients count. The difference between the average delay values of different real-life simulations showed that with increasing number of clients SSPayWMN protocols show higher network delays. However, the increase is linear; therefore, SSPayWMN ensures stable performance in different sized networks.

6. CONCLUSIONS

In this thesis, we have proposed a secure and seamless prepayment scheme for wireless mesh networks (SSPayWMN). Our payment scheme is based on hash chains and provides undeniable payment. This system provides fairness to both operators and to clients. Moreover, SSPayWMN provides privacy and untraceability to some extent. Furthermore; SSPayWMN can successfully handle seamless handover between operators by eliminating the need for re-authentication from the scratch.

We have conducted two types of simulations to evaluate our system using ns-3: Unit tests and real-life scenario simulations. In unit tests we have evaluated the standalone performances of the protocols. In real-life scenario simulations, we have introduced client types and their mobility schemes to achieve realistic simulation results. The real-life scenario simulations evaluate the system performance for an ordinary day usage.

Our results for unit tests show that SSPayWMN protocols achieve steady state performances in independent executions of system protocols. Real-life scenario simulations show that SSPayWMN protocols could achieve steady state performance in a daily basis. Both simulations ensure the stable performance of SSPayWMN and shows that our system is a flexible and robust system.

REFERENCES

[1] Akyildiz, I.F. Wang, X. and Wang, W. (2005). Wireless Mesh Networks: A Survey. Computer Networks and ISDN Systems, pp. 445-487.

[2] Vaughan-Nichols S.J. (2004). Achieving Wireless Broadband with WiMax, IEEE Computer, vol. 37, no. 6, pp. 10-13.

[3] The ZigBee Alliance. Retrieved from http://www.zigbee.org

[4] Rao, Y.S. Yeung W.C. and Kripalani, A. (2000). Third-generation (3G) Radio Access Standards, Communication Technology Proceedings, 2000. WCC - ICCT 2000 International Conference, vol.2, pp. 1017-1023.

[5] Rivest, R.L. and Shamir, A. (2001). Pay-word and MicroMint: Two Simple Micropayment Schemes. International Journal of Network Security, pp. 81-90.

[6] Al-Fayoumi, M.A. Aboud, S.J. Al-Fayoumi, M.A. and AlgHazzawi, D. (2010). Efficient E-Payment Protocol Using Hash Chain. European, Mediterranean & Middle Eastern Conference on Information Systems 2010.

[7] Qi, J. Zhao, Y. Wang, X. Choi J. (2010). Security Authentication and an Undeniable Billing Protocol for WMNs. International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, pp. 266-269.

[8] Yucel, C. (2010). A Secure Prepaid Micropayment Scheme for Wireless Mesh Networks. MS Thesis, Sabanci University, Istanbul, TR.

[9] Trappe, W. and Washington, L.C. (2005). Introduction to Cryptography with Coding Theory. Prentice Hall.

[10] Rivest, R. (1992). The MD5 Message-Digest Algorithm, MIT Laboratory of Computer Science and RSA Data Security, Inc.

[11] FIPS PUB 180-1, (1995). Secure Hash Standard, Retrieved from http://www.itl.nist.gov/fipspubs/fip180-1.htm.

[12] FIPS PUB 180-2, (2002). Secure Hash Standard, Retrieved from http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf.

[13] Bertoni, G. Daemen, J. Peeters. M. and Assche, G.V. (2011). The KECCAK Sponge Function Family, Retrieved from http://keccak.noekeon.org/

[14] Lamport, L. (1981). Password Authentication with Insecure Communication. Commun. ACM., vol. 24, no. 11, pp. 770-772.

[15] Stallings, W. (2011). Cryptography and Network Security: Principles and Practices, 5th edition, Prentice Hall, NJ.

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

[17] Robshaw, M.J.B. (1995). Stream Ciphers, RSA Laboratories Technical Report, Lecture Notes in Computer Science.

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

[19] Rivest, R.L. (1994). The RC5 Encryption Algorithm. Proceedings of the Second International Workshop on Fast Software Encryption (FSE), pp. 86–96.

[20] Schneier, B. (1993). Description of a New Variable Length Key, 64 bit Block Cipher (Blowfish), Fast Software Encryption, Cambridge Security Workshop Proceedings, pp. 191-204.

[21] Schneier, B. Kelsey, J. Whiting, D. Wagner, D. Hall, C. and Ferguson, N. (1999). Performance Comparison of the AES Submissions. Second AES Candidate Conference, pp. 15-34.

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

[23] Miller, V. (1985). Use of Elliptic Curves in Cryptography, Advances in Cryptology - CRYPTO 1985, Lecture Notes in Computer Science, Springer-Verlag, vol. 218, pp. 417–426.

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

[25] Elgamal, T. (1985). A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Transactions on Information Theory, vol 31, no. 4, pp. 469-472.

[26] Rivest, R. Shamir, A. and Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, vol. 21, pp. 120–126.

[27] Tiwari, A. Sanyal, S. Abraham, A. Knapskog, J.S. and Sanyal, S. (2007). A Multi-factor Security Protocol for Wireless Payment-Secure Web Authentication Using Mobile Devices. IADIS International Conference Applied Computing, pp. 160-167.

[28] Bayyapu, P.R. and Das, M.L. (2008). An Improved and Efficient Micro-Payment Scheme, Retrieved from http://www.jtaer.com.

[29] Chien H.J.J. and Tseng Y. (2001). RSA-based Partially Blind Signature with Low Computation. Proceeding of the International Conference in Parallel and Distributed Systems, pp. 385–389.

[30] Hwang, M.S. and Sung, P.C. (2006). A Study of Micro-payment Based on One-way Hash Chain. International Journal of Network Security, vol. 2, no.2, pp. 81-90.

[31] Chen, Y.Y. Jan, J.K. and Chen, C.L. (2005). A Fair and Secure Mobile Billing System. Computer Networks, vol. 48, no.4, pp. 517-524.

[32] Li, S. Wang, G. Zhou, J. and Chen, K. (2007). Undeniable Mobile Billing Schemes. European Public Key Infrastructure Workshop - EUROPKI, pp. 338-345.

[33] Zaghloul, S. Bziuk, W. and Jukan, A. (2008). A Scalable Billing Architecture for Future Wireless Mesh Backhauls, IEEE ICC '08.

[34] Zhang, Y. and Fang, Y. (2007). A Secure Authentication and Billing Architecture for Wireless Mesh Networks, Wireless Networks, vol. 13, no. 5, pp. 663-678.

[35] Qi, J. Zhao, Y. Wang, X. and Choi, J. (2010). Security Authentication and an Undeniable Billing Protocol for WMNs. International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, pp. 266-269.

[36] Johnson, D.B. and Maltz, D.A. (1996). Dynamic Source Routing in Ad Hoc Wirelees Networks In Mobile Computing, In Computer Communications Review, Proceedings of SIGCOMM '96.

[37] Crescenzi, P. Ianni, M.D. Marino, A. Rossi, G. and Vocca, P. (2009). Spatial Node Distribution of Manhattan Path Based Random Waypoint Mobility Models with Applications. Sirocco 2009, pp. 154-166.

[38] OMNET++ Discrete Event Simulator, Retrieved from http://www.omnetpp.org

[39] Network Simulator 3 (ns-3), Retrieved from http://www.nsnam.org

[40] Yang, K. Ma, J. and Miao, Z. (2009). Hybrid Routing Protocol for Wireless Mesh Network, Computational Intelligence and Security – CIS '09.

[41] Chaum, D. (1981). Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM, vol. 24, no.2, pp. 84-90.

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

[43] Deng, L. and Kuzmanovic, A. (2009). A Feeder-carrier-based Internet User Accountability Service, Special Interest Group on Data Communication (SIGCOMM) 2009.

[44] OpenSSL Speed on iPhone4. Retrieved from http://hmijailblog.blogspot.com/2011/02/openssl-speed-on-iphone-4.html

[45] Crypto++ 5.6.0 Benchmarks. Retrieved from http://www.cryptopp.com/benchmarks.html

APPENDIX 1

We have considered the clients of SSPayWMN to have iPhone 4 on their hands. The system causes network delays as well as computational delays.

OpenSSL is run to understand that how much time an iPhone 4 needs to finish a certain cryptographic task. OpenSSL is run on a MacBook by connecting the iPhone 4 to it. The result of the command "opensll speed" is presented below.

The highlighted values of the result are used.

OpenSSL run on iPhone 4:

```
Doing md2 for 3s on 16 size blocks: 113903 md2's in 2.93s
Doing md2 for 3s on 64 size blocks: 59789 md2's in 2.96s
Doing md2 for 3s on 256 size blocks: 20740 md2's in 2.94s
Doing md2 for 3s on 1024 size blocks: 5722 md2's in 2.95s
Doing md2 for 3s on 8192 size blocks: 741 md2's in 2.95s
Doing md4 for 3s on 16 size blocks: 962184 md4's in 2.99s
Doing md4 for 3s on 64 size blocks: 808365 md4's in 2.94s
Doing md4 for 3s on 256 size blocks: 553565 md4's in 2.94s
Doing md4 for 3s on 1024 size blocks: 251644 md4's in 2.96s
Doing md4 for 3s on 8192 size blocks: 41204 md4's in 2.94s
Doing md5 for 3s on 16 size blocks: 512656 md5's in 2.88s
Doing md5 for 3s on 64 size blocks: 446107 md5's in 2.70s
Doing md5 for 3s on 256 size blocks: 323702 md5's in 2.65s
Doing md5 for 3s on 1024 size blocks: 169849 md5's in 2.79s
Doing md5 for 3s on 8192 size blocks: 30073 md5's in 2.68s
Doing hmac(md5) for 3s on 16 size blocks: 933276 hmac(md5)'s in 2.62s
Doing hmac(md5) for 3s on 64 size blocks: 741506 hmac(md5)'s in 2.72s
Doing hmac(md5) for 3s on 256 size blocks: 490370 hmac(md5)'s in 2.62s
Doing hmac(md5) for 3s on 1024 size blocks: 202907 hmac(md5)'s in 2.63s
Doing hmac(md5) for 3s on 8192 size blocks: 28168 hmac(md5)'s in 2.32s
Doing sha1 for 3s on 16 size blocks: 466460 sha1's in 2.83s
Doing sha1 for 3s on 64 size blocks: 498797 sha1's in 2.92s
Doing sha1 for 3s on 256 size blocks: 298535 sha1's in 2.90s
Doing sha1 for 3s on 1024 size blocks: 114257 sha1's in 2.94s
Doing sha1 for 3s on 8192 size blocks: 16773 sha1's in 2.94s
Doing sha256 for 3s on 16 size blocks: 577481 sha256's in 2.93s
Doing sha256 for 3s on 64 size blocks: 350266 sha256's in 2.96s
Doing sha256 for 3s on 256 size blocks: 158648 sha256's in 2.96s
Doing sha256 for 3s on 1024 size blocks: 49984 sha256's in 2.93s
Doing sha256 for 3s on 8192 size blocks: 6722 sha256's in 2.88s
Doing sha512 for 3s on 16 size blocks: 120505 sha512's in 2.93s
Doing sha512 for 3s on 64 size blocks: 121406 sha512's in 2.91s
Doing sha512 for 3s on 256 size blocks: 43968 sha512's in 2.92s
Doing sha512 for 3s on 1024 size blocks: 15084 sha512's in 2.94s
Doing sha512 for 3s on 8192 size blocks: 2119 sha512's in 2.96s
Doing rmd160 for 3s on 16 size blocks: 500321 rmd160's in 2.88s
Doing rmd160 for 3s on 64 size blocks: 518314 rmd160's in 2.93s
Doing rmd160 for 3s on 256 size blocks: 294776 rmd160's in 2.93s
Doing rmd160 for 3s on 1024 size blocks: 108497 rmd160's in 2.92s
Doing rmd160 for 3s on 8192 size blocks: 14321 rmd160's in 2.69s
Doing rc4 for 3s on 16 size blocks: 8838055 rc4's in 2.96s
Doing rc4 for 3s on 64 size blocks: 2573509 rc4's in 2.91s
Doing rc4 for 3s on 256 size blocks: 670210 rc4's in 2.93s
Doing rc4 for 3s on 1024 size blocks: 168703 rc4's in 2.92s
Doing rc4 for 3s on 8192 size blocks: 20854 rc4's in 2.71s
Doing des cbc for 3s on 16 size blocks: 1656140 des cbc's in 2.75s
Doing des cbc for 3s on 64 size blocks: 425114 des cbc's in 2.83s
Doing des cbc for 3s on 256 size blocks: 108385 des cbc's in 2.87s
Doing des cbc for 3s on 1024 size blocks: 28718 des cbc's in 2.76s
Doing des cbc for 3s on 8192 size blocks: 3510 des cbc's in 2.81s
Doing des ede3 for 3s on 16 size blocks: 646206 des ede3's in 2.83s
Doing des ede3 for 3s on 64 size blocks: 164355 des ede3's in 2.87s
Doing des ede3 for 3s on 256 size blocks: 41628 des ede3's in 2.83s
Doing des ede3 for 3s on 1024 size blocks: 10274 des ede3's in 2.87s
Doing des ede3 for 3s on 8192 size blocks: 1241 des ede3's in 2.80s
Doing aes-128 cbc for 3s on 16 size blocks: 2403683 aes-128 cbc's in 2.89s
Doing aes-128 cbc for 3s on 64 size blocks: 649635 aes-128 cbc's in 2.81s
Doing aes-128 cbc for 3s on 256 size blocks: 174904 aes-128 cbc's in 2.89s
Doing aes-128 cbc for 3s on 1024 size blocks: 45134 aes-128 cbc's in 2.94s
Doing aes-128 cbc for 3s on 8192 size blocks: 5175 aes-128 cbc's in 2.68s
Doing aes-192 cbc for 3s on 16 size blocks: 2168292 aes-192 cbc's in 2.91s
Doing aes-192 cbc for 3s on 64 size blocks: 600481 aes-192 cbc's in 2.93s
Doing aes-192 cbc for 3s on 256 size blocks: 155498 aes-192 cbc's in 2.90s
Doing aes-192 cbc for 3s on 1024 size blocks: 39192 aes-192 cbc's in 2.91s
Doing aes-192 cbc for 3s on 8192 size blocks: 4777 aes-192 cbc's in 2.86s
Doing aes-256 cbc for 3s on 16 size blocks: 1983041 aes-256 cbc's in 2.94s
Doing aes-256 cbc for 3s on 64 size blocks: 540276 aes-256 cbc's in 2.90s
Doing aes-256 cbc for 3s on 256 size blocks: 137713 aes-256 cbc's in 2.89s
Doing aes-256 cbc for 3s on 1024 size blocks: 35393 aes-256 cbc's in 2.93s
Doing aes-256 cbc for 3s on 8192 size blocks: 4384 aes-256 cbc's in 2.93s
Doing aes-128 ige for 3s on 16 size blocks: 2375637 aes-128 ige's in 2.92s
Doing aes-128 ige for 3s on 64 size blocks: 698695 aes-128 ige's in 2.94s
Doing aes-128 ige for 3s on 256 size blocks: 183967 aes-128 ige's in 2.91s
Doing aes-128 ige for 3s on 1024 size blocks: 46891 aes-128 ige's in 2.86s
Doing aes-128 ige for 3s on 8192 size blocks: 5870 aes-128 ige's in 2.93s
Doing aes-192 ige for 3s on 16 size blocks: 2129667 aes-192 ige's in 2.94s
Doing aes-192 ige for 3s on 64 size blocks: 615957 aes-192 ige's in 2.90s
Doing aes-192 ige for 3s on 256 size blocks: 159366 aes-192 ige's in 2.89s
Doing aes-192 ige for 3s on 1024 size blocks: 40431 aes-192 ige's in 2.90s
Doing aes-192 ige for 3s on 8192 size blocks: 4566 aes-192 ige's in 2.62s
Doing aes-256 ige for 3s on 16 size blocks: 1874993 aes-256 ige's in 2.85s
Doing aes-256 ige for 3s on 64 size blocks: 546698 aes-256 ige's in 2.93s
Doing aes-256 ige for 3s on 256 size blocks: 143584 aes-256 ige's in 2.87s
Doing aes-256 ige for 3s on 1024 size blocks: 36373 aes-256 ige's in 2.82s
Doing aes-256 ige for 3s on 8192 size blocks: 4479 aes-256 ige's in 2.88s
Doing idea cbc for 3s on 16 size blocks: 1616991 idea cbc's in 2.84s
Doing idea cbc for 3s on 64 size blocks: 419827 idea cbc's in 2.85s
Doing idea cbc for 3s on 256 size blocks: 107747 idea cbc's in 2.91s
Doing idea cbc for 3s on 1024 size blocks: 26694 idea cbc's in 2.90s
Doing idea cbc for 3s on 8192 size blocks: 3366 idea cbc's in 2.91s
Doing rc2 cbc for 3s on 16 size blocks: 1456945 rc2 cbc's in 2.86s
Doing rc2 cbc for 3s on 64 size blocks: 387618 rc2 cbc's in 2.92s
Doing rc2 cbc for 3s on 256 size blocks: 98467 rc2 cbc's in 2.89s
Doing rc2 cbc for 3s on 1024 size blocks: 24847 rc2 cbc's in 2.92s
Doing rc2 cbc for 3s on 8192 size blocks: 3091 rc2 cbc's in 2.95s
Doing blowfish cbc for 3s on 16 size blocks: 3588480 blowfish cbc's in 2.92s
Doing blowfish cbc for 3s on 64 size blocks: 969771 blowfish cbc's in 2.92s
Doing blowfish cbc for 3s on 256 size blocks: 244745 blowfish cbc's in 2.89s
Doing blowfish cbc for 3s on 1024 size blocks: 63589 blowfish cbc's in 2.97s
Doing blowfish cbc for 3s on 8192 size blocks: 7246 blowfish cbc's in 2.65s
Doing cast cbc for 3s on 16 size blocks: 2605484 cast cbc's in 2.93s
Doing cast cbc for 3s on 64 size blocks: 689905 cast cbc's in 2.88s
Doing cast cbc for 3s on 256 size blocks: 158869 cast cbc's in 2.80s
Doing cast cbc for 3s on 1024 size blocks: 44980 cast cbc's in 2.90s
Doing cast cbc for 3s on 8192 size blocks: 6318 cast cbc's in 2.92s
Doing 512 bit private rsa's for 10s: 2657 512 bit private RSA's in 9.77s
Doing 512 bit public rsa's for 10s: 29891 512 bit public RSA's in 9.75s
Doing 1024 bit private rsa's for 10s: 507 1024 bit private RSA's in 9.81s
Doing 1024 bit public rsa's for 10s: 9986 1024 bit public RSA's in 9.80s
Doing 2048 bit private rsa's for 10s: 79 2048 bit private RSA's in 9.53s
Doing 2048 bit public rsa's for 10s: 2934 2048 bit public RSA's in 9.85s
Doing 4096 bit private rsa's for 10s: 13 4096 bit private RSA's in 10.50s
Doing 4096 bit public rsa's for 10s: 818 4096 bit public RSA's in 9.70s
Doing 512 bit sign dsa's for 10s: 3096 512 bit DSA signs in 9.80s
Doing 512 bit verify dsa's for 10s: 2681 512 bit DSA verify in 9.78s
Doing 1024 bit sign dsa's for 10s: 1004 1024 bit DSA signs in 9.58s
Doing 1024 bit verify dsa's for 10s: 877 1024 bit DSA verify in 9.78s
Doing 2048 bit sign dsa's for 10s: 297 2048 bit DSA signs in 9.68s
Doing 2048 bit verify dsa's for 10s: 249 2048 bit DSA verify in 9.73s
OpenSSL 0.9.8k 25 Mar 2009
built on: date not available
options:bn(64,32) md2(int) rc4(ptr,char) des(idx,cisc,16,long) aes(partial) idea(int)
blowfish(ptr)
compiler: arm-apple-darwin9-gcc -fPIC -fno-common -DOPENSSL_PIC -DOPENSSL_THREADS -D_REENTRANT
-DDSO_DLFCN -DHAVE_DLFCN_H -D__DARWIN_UNIX03 -O3 -fomit-frame-pointer -fno-common
available timing options: TIMEB USE_TOD HZ=100 [sysconf value]
timing function used: getrusage
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes
md2                622.00k     1292.74k     1805.93k     1986.21k     2057.72k
mdc2                 0.00         0.00         0.00         0.00         0.00
md4               5148.81k    17597.06k    48201.58k    87055.22k   114810.60k
md5               2848.09k    10574.39k    31270.83k    62338.84k    91924.63k
hmac(md5)         5699.40k    17447.20k    47914.02k    79002.57k    99462.18k
sha1              2637.23k    10932.54k    26353.43k    39795.64k    46736.20k
rmd160            2779.56k    11321.53k    25755.17k    38048.26k    43612.50k
rc4              47773.27k    56599.51k    58557.60k    59161.60k    63039.10k
```

```
des cbc           9635.72k     9613.89k     9667.79k    10654.79k    10232.71k
des ede3          3653.46k     3665.06k     3765.64k     3665.71k     3630.81k
idea cbc          9109.81k     9427.69k     9478.77k     9425.74k     9475.69k
seed cbc             0.00         0.00         0.00         0.00         0.00
rc2 cbc           8150.74k     8495.74k     8722.34k     8713.47k     8583.55k
rc5-32/12 cbc        0.00         0.00         0.00         0.00         0.00
blowfish cbc     19662.90k    21255.25k    21679.83k    21924.29k    22399.71k
cast cbc         14227.90k    15331.22k    14525.17k    15882.59k    17725.02k
aes-128 cbc      13307.59k    14795.96k    15493.23k    15720.14k    15818.51k
aes-192 cbc      11921.88k    13116.31k    13726.72k    13791.27k    13682.93k
aes-256 cbc      10792.06k    11923.33k    12198.80k    12369.43k    12257.25k
camellia-128 cbc        0.00         0.00         0.00         0.00         0.00
camellia-192 cbc        0.00         0.00         0.00         0.00         0.00
camellia-256 cbc        0.00         0.00         0.00         0.00         0.00
sha256            3153.48k     7573.32k    13720.91k    17468.81k    19120.36k
sha512             658.05k     2670.10k     3854.73k     5253.75k     5864.48k
aes-128 ige      13017.19k    15209.69k    16184.04k    16788.95k    16411.96k
aes-192 ige      11590.02k    13593.53k    14116.85k    14276.33k    14276.59k
aes-256 ige      10526.28k    11941.53k    12807.49k    13207.78k    12740.27k
                  sign    verify    sign/s verify/s
rsa  512 bits 0.003677s 0.000326s    272.0   3065.7
rsa 1024 bits 0.019349s 0.000981s     51.7   1019.0
rsa 2048 bits 0.120633s 0.003357s      8.3    297.9
rsa 4096 bits 0.807692s 0.011858s      1.2     84.3
                  sign    verify    sign/s verify/s
dsa  512 bits 0.003165s 0.003648s    315.9    274.1
dsa 1024 bits 0.009542s 0.011152s    104.8     89.7
dsa 2048 bits 0.032593s 0.039076s     30.7     25.6
```