secure and seamless payment 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  
December 2012

SECURE AND SEAMLESS PAYMENT FOR WIRELESS MESH NETWORKS

APPROVED BY

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

(Thesis Supervisor)

DATE OF APPROVAL ........................................

© Serhat Can Leloğlu 2012

All Rights Reserved

SECURE AND SEAMLESS PAYMENT FOR WIRELESS MESH NETWORKS

Serhat Can Leloğlu  
Computer Science and Engineering, MS Thesis, 2012  
Thesis Supervisor: Assoc. Prof. Albert Levi  
Keywords: Prepaid Payment Systems, Security, Micropayments, Wireless Mesh Networks

Abstract

Wireless Mesh Network (WMN) is a multi-hop high-speed networking technology for broadband network access. Compared to conventional service providing systems it is fairly cost-effective and easy to deploy. WMNs are often used for service providing since they provide service to mobile and immobile clients. In this paper a secure and seamless pre-payment system is proposed moreover network simulations for this system are presented.

In the WMN structure that we assume in SSPayWMN project there are mobile clients and operators, who will be charging the service they give. We assume there is more than one operator and users should be able to get service from these operators. In case of a roaming situation, service should not be interrupted and users should continue getting service without noticing operator change has occurred. Related works for broadband access usually trust operators fully, but in real life operators may unintentionally overcharge their users and these cause disputes between the customers and the operators. Even in the cases where the operator is right, it is very difficult to convince the customer since the operators generally do not have justifiable proofs that cannot be denied by the customers.

In SSPayWMN project, our aim is to design a secure payment scheme, which is fair to both the operator and users. Using cryptographic tools and techniques, all system parties will make sure that they talk to the correct entity and providing/getting service in an undeniable way. We will design and implement a secure prepaid payment scheme and we will prove our system’s effectiveness by implementing our system on a network simulator. By doing so, we aim to get near real life performance results for critical use cases.

We will design our system considering our main requirements. Our requirements are to be studied under two categories. One of them is network requirements and the other one is cryptographic requirements. They are briefly explained in following sections.

KABLOSUZ ÖRGÜ AĞLARI İÇİN GÜVENLİ VE KESİNTİSİZ ÖN ÖDEMELİ MİKROÖDEME

Serhat Can Leloğlu  
Bilgisayar Bilimi ve Mühendisliği, Yüksek Lisans Tezi, 2012  
Tez Danışmanı: Doç. Dr. Albert Levi  
Anahtar Kelimeler: Ön Ödemeli Sistemler, Güvenlik, Mikro Ödeme, Kablosuz Örgü Ağlar

Özet

Kablosuz Örgü Ağları (KÖA) çok sekmeli, erişimin her yerden sağlanabildiği, yüksek hızlı ve gelişmekte olan bir ağ teknolojisidir. Kablosuz Örgü Ağları ağ yapısında esneklik sağlar ve kendiliğinden kurulabilen dinamik bir yapısı vardır. Bu ağlardaki elemanlar rota tabloları oluşturabilirler ve yüksek hızda iletişim sağlayabilirler. Kablosuz Örgü Ağları kurulması kolay sistemlerdir ve ekstra yükleri azdır.

Bizim KÖA sistemimizde mobil halde kullanıcılar ve verdikleri hizmete göre ücret talep eden operatörler var. Varsayımımıza göre birden fazla operatör olacak ve kullanıcılar istedikleri operatörlerden hizmet almakta özgür olacaklar. Operatör değişikliği olduğu durumlarda servis kesintisiz şekilde devam edecek ve kullanıcı operatör değiştirdiğinin bile farkına varmayacak. Bu alanda yapılan diğer çalışmalar genel olarak operatörlere sonsuz güven ilkesine dayanarak tasarlanmıştır. SSPayWMN projesinde biz operatörlere mutlak güven ilkesini benimsemedik ve sistemimizi bu doğrultuda kurduk. Bunun nedeni operatörlerin istem dışı ücret almalarını engellemek. SSPayWMN sisteminde bir kullanıcı parasını ödemediği hiç bir hizmeti alamaz aynı zamanda kimse kullanıcının almadığı hizmetin parasını da alamaz. Bu proje operatör için de kullanıcı için de isteklerin kriptografik yollarla ispatlanmasına olanak tanıyor.

SSPayWMN projesinde amacımız hem operatörler hem de kullanıcılar için adil bir sistem kurmatı. Kriptografik araçlar ve algoritmalar kullanarak bütün ağ elemanları kullandıkları servisi ve ödedikleri ücerti ispatlayabilme şansına sahipler. Kurduğumuz sistemin güvenli ve hızlı olduğunu bu dökümanda anlatıyoruz ve efektif çalıştığını görebilmek için network simulasyonı sonuçlarını da paylaşıyoruz. Bu sayede gerçek hayattaki uygulamaya çok yakın sonuçlar elde etmiş oluyoruz.

Sistemimizi ana gerekliliklere göre kurduk. Bu gereklilikler iki dalda incelenebilir. Bu alanlar ağ gereklilikleri ve kriptografik gereklilikler. Bu dallar dökümanın ilerleyen sayfalarında açıklanacaklar.

*To my dear family*

Contents

[1. INTRODUCTION 13](#_Toc343721020)

[1.1 Contribution of the Thesis 13](#_Toc343721021)

[1.2 Organization of the Thesis 15](#_Toc343721022)

[2. Background on wireless mesh networks 16](#_Toc343721023)

[2.1 Components of WMN 16](#_Toc343721024)

[2.2 Characteristics 18](#_Toc343721025)

[3. Background on Cryptographic functions 19](#_Toc343721026)

[3.1 Hash Functions 19](#_Toc343721027)

[3.2 Hash Chains 20](#_Toc343721028)

[3.3 HMAC 21](#_Toc343721029)

[3.4 Symmetric Cryptography 22](#_Toc343721030)

[3.5 Public Key Cryptography 23](#_Toc343721031)

[4. Requirements for a secure and seamless mıcropayment scheme ın wıreless mesh networks 25](#_Toc343721032)

[4.1 Requirements of the Network 25](#_Toc343721033)

[4.2 General Overview of the Proposed Scheme 26](#_Toc343721034)

[4.3 Connection Card Structure 29](#_Toc343721035)

[4.4 Alias Computation 30](#_Toc343721036)

[5. Protocols of secure and seamless mıcropayment scheme 32](#_Toc343721037)

[5.1 Initial Authorization 32](#_Toc343721038)

[5.2 Reuse of a Connection Card 34](#_Toc343721039)

[5.3 Access Point Authentication 36](#_Toc343721040)

[5.4 Packet Transfer 37](#_Toc343721041)

[5.5 Changing Alias 39](#_Toc343721042)

[5.6 Update Packets 41](#_Toc343721043)

[5.7 Disconnection 43](#_Toc343721044)

[5.8 Distributing Access Point Public Keys 44](#_Toc343721045)

[5.9 Seamless Roaming (Payment Related) 46](#_Toc343721046)

[5.10 Seamless Mobility in Home Operator (Payment Related) 48](#_Toc343721047)

[6. Unit test results 50](#_Toc343721048)

[6.1 Unit Test Result for End-to-End Two-Way Protocols 50](#_Toc343721049)

[6.2 Unit Test Result for Access Point Authentication 51](#_Toc343721050)

[6.3 Unit Test Result for Seamless Mobility and Roaming 51](#_Toc343721051)

[6.4 Unit Test Result for Packet Transfer 52](#_Toc343721052)

[6.5 Unit Test Result for Update Packets 53](#_Toc343721053)

[7. User Modeling And Mobility 54](#_Toc343721054)

[7.1 User Actions 54](#_Toc343721055)

[7.2 Client Types 55](#_Toc343721056)

[7.3 User Mobility and Timing 57](#_Toc343721057)

[8. Results for Real-Life Scenario Simulation 59](#_Toc343721058)

[8.1 Overview 59](#_Toc343721059)

[8.2 Real-Life Scenario Simulation Result for Initial Authorization 60](#_Toc343721060)

[8.3 Real-Life Scenario Simulation Result for Reuse of a Connection Card Protocol 61](#_Toc343721061)

[8.4 Real-Life Scenario Simulation Result for Changing Alias 62](#_Toc343721062)

[8.5 Real-Life Scenario Simulation Result for Disconnection 62](#_Toc343721063)

[8.6 Real-Life Scenario Simulation Result for Update Packets 63](#_Toc343721064)

[8.7 Real-Life Scenario Simulation Result for Seamless Mobility in Home Operator Protocol 63](#_Toc343721065)

[8.8 Real-Life Scenario Simulation Result for Roaming Protocol 64](#_Toc343721066)

[8.9 Real-Life Scenario Simulation Result for Packet Transfer 64](#_Toc343721067)

[9. Conclusion 66](#_Toc343721068)

Acknowledgements

**LIST OF FIGURES**

**LIST OF TABLES**

# INTRODUCTION

Wireless Mesh Networks [1] are often used for service providing; moreover a secure system built using WMNs should support user identification, authentication as well as authorization and accounting.

Commonly payment systems service providers do not fully trust clients, but in reality service providers –intentionally or not- may over charge the clients or charge for services that they did not provide. It is proven that using native cryptographic algorithms, every action could have an undeniable cryptographic proof so that the client could not get service without payment and service providers could not charge without serving.

The secure and seamless pre-payment system presented in this paper, has the properties such as wide-coverage, seamless mobility and roaming, anonymity, mutual authentication, two-way honesty, preventing double spending and unlinkability. Ten protocols are designed for actions of the system entities, and these protocols are tested using network simulator 3 (ns-3) [6]. The designed system had formidable results in unit tests and the results are explained in this paper too.

## Contribution of the Thesis

Authentication, confidentiality, non-repudiation, fraudprotectionwill be provided in oursystem. The users will not be able to deny using credits for the services actually obtained; the operator will not be able to charge more than the usage amount. Moreover, inter-operator settlement will be performed in a secure way such that each operator will have cryptographic proofs of use for the services that they provide to other operators' customers. In order to provide privacy of individuals, our scheme will provide untraceability such that no unauthorized entity will be able to track down a particular user.

Since the clients are mobile, they may hand over 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 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.

## Organization of the Thesis

The organization of the thesis is as follows…

# BACKGROUND ON WIRELESS MESH NETWORKS

Wireless Mesh Network (WMN) is multi-hop wireless networking type, designed as an alternative to traditional centralized wireless networking achieved by mesh routers [1]. Mesh routers and mesh 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 postulate improved scalability by optimizing the medium access control (MAC) protocol in a mesh router.

## 2.1 Network Architecture

Three main groups depending on operation of the nodes could accomplish the categorization of WMNs.

*Infrastructure/Backbone WMNs:* The architecture is shown in Figure 1. Both wireless and wired networks comprise infrastructure WMNs, in Fig. 1 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 1. Infrastructure/backbone WMNs. [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 1 base stations could also connect to mesh backbones, which provides Internet connectivity for all the clients of base stations.

*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 whomever wants to join to the network has to be a part of the backbone and provide routing functionality. As shown in Figure 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 whereas they can communicate within each other by using a simple ad-hoc network.



Figure 3. Hybrid WMNs [1]

As shown in Figure 3 mesh backbone provides Internet connectivity whereas Client WMNs provide connectivity to mesh backbone for far located mesh clients.

## 2.2 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 the abilities like self-configuring and self-healing. Deployment, node addition and removal are easy to accomplish since mesh routers form 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 peer-to-peer protocols.

*Dependence of Power-Consumption Constraints on the Type of Mesh Nodes:* Mesh routers do not have power-consumption constraintsin common but it is advisable for mesh clients to have some forms of power consumption constraints.

*Compatibility and Interoperability with Existing Wireless Networks:* WMNs are compatible with IEEE 802.11 protocols [2,3], therefore WMNs could support for both mesh purposes and also conventional Wi-Fi connections. WiMax [4], ZigBee [5] and cellular networks could also inter-connect with WMN structure.

# Background ON Cryptographic algorithms

To establish a secure system, cryptographic primitive algorithms are employed. A brief explanation and introduction for cryptographic primitives are provided in this thesis to provide unity in the document.

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

## 1.1. Hash Functions

Hash functions [7] are irreversible mathematical functions that map input strings of variable length to fixed sized output strings. Hash functions are usually employed for improving time performance of table lookup or data comparison tasks such as finding items in a database, discovering repeated or analogous records in a bulky file, finding similar springs in a DNA string and cryptographic purposes.



Figure 4. Hash Function Example [7]

Figure 4 depicts the hash function flow. The function maps a longer message to a 160-bit bit string. The output length depends on the hash algorithm; various hash algorithms have different output sizes.

Hash functions could receive various sized parameters but generate fixed sized input strings. Compared to mainstream cryptographic algorithms, hash functions are fairly cost-effective in both power and time consumption. Light-weightiness of hash functions make them eligible for security systems.

A hash function should satisfy following properties:

1. Given a message m, the message digest h(m) can be calculated very quickly.
2. Given a y, it is computationally infeasible to find an relation with h (m'). =*Y* (in other words, *h* is a one-way, or collision resistant, function)

The most popular and well-regarded hash functions are MD5 [24], SHA1 [23] and SHA2 [23], which is a set of SHA-224, SHA-256, SHA-384 and SHA-512.

## 1.2 Hash Chains

Applying a hash algorithm to an initial value and using the output as an input for the next hash function forms a hash chain. Every output of a hash algorithm represents a link in the chain. Length of the hash chain is determined by the number of times the hash algorithm is executed.



Figure 5. Hash Chain Depiction and Usage [8]

Since hash functions are irreversible, as shown in Figure 5 it is easy to go forward in the chain but it is not computationally feasible to go backwards. Which means a person could find any value on the chain if she knows the initial value but this situation is not possible for a person who knows the last value on the chain.

A hash chain with 5 elements is denoted as:

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

n times

Because of the fact that hash functions are one-way mathematical functions, it is appropriate to say that hash functions are good tools for security systems communicating through insecure links. Knowing the first link in the chain 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 or secrets for other cryptographic functions.

Hash chains are easy-to-deploy and cost-effective therefore they are widely used in cryptographic systems. Especially for systems that have delicacy for computational delay hash chains are effective tools.

## 1.3 HMAC Functions

One of the main research areas in cryptography and computer networking is providing integrity and reliability on a transmitted or stored data. Message Authentication Codes (MACs) are use shared secret keys; therefore they are suitable for such integrity checks.

Classically, MACs are used between two parties that share a secret key in order to authenticate the transmitted or stored data between these parties. This protocol executes a MAC that uses a cryptographic hash function union with a secret key.

HMACs are used together with widely accepted hash functions. HMAC employs a secret key for generation and verification of the MACs. The aims of HMAC construction [9] are:

1. Using hash functions without making any changes on them. Previously implemented codes and hardware shall work with the deployment of HMAC.
2. Maintaining the original fastness of the hash functions.
3. Using and handling secret keys in a cost-effective way.
4. Providing provable and reasonable cryptographic analyzes using the previously done performance analysis of underlying hash functions.
5. Achieving faster and more robust performances in a case of a faster hash function is invented in the future. Replacement should be easy-to-achieve.

Table 1.1 explains the parameters HMAC uses.

Table 1.1. HMAC Parameters [9]

|  |  |
| --- | --- |
| B | Block size in bytes |
| H | A secure and fast hash function |
| ipad | Inner pad, the byte x36 B times |
| K | Shared secret key |
| K0 | The key K before any process to make it B bytes long |
| L | Block size of the output of the hash function, in bytes |
| opad | Outer pad, the byte x5c repeated B times |
| t | The number of bytes of MAC. |
| text | The data that used to calculate HMAC |
| xN | Hexadecimal notation where each string N represents 4 binary bits |
|  | Exclusive-Or operation |
|  | Concatenation |

Figure 6 shows the steps of HMAC.



Figure 6. Steps of HMAC [9]

## 1.4 Symmetric Cryptography

Symmetric cryptography is the oldest kind of cryptographic primitive. This primitive employs shared secret keys between two parties. The security level of a symmetric cryptographic algorithm mostly depends on key size. Modern algorithms use at least 128-bit long keys.



Figure 7. Symmetric Cryptography Scheme [16]

Symmetric key cryptography employs a secret key between two parties. As shown in Figure 7 a plaintext input is used as a parameter with the shared secret key in a encryption algorithm. Superficially the encryption and decryption algorithms are black boxes from the parties’ point of view. Encrypted data is transmitted through a insecure medium. The receiver of the encrypted message decrypts the cipher text with the shared secret key and calculates the original message.

Modern symmetric cryptographic functions could be categorized under two classes, which are stream ciphers and block ciphers. Stream ciphers encrypt data byte by byte. The most widely used stream cipher is RC4 [15]. Secure Socket Layer (SSL) and Wired Equivalent Privacy (WEP) employs RC4. On the other hand block ciphers encrypt an input data as fixed size blocks, and produces same-sized outputs. The most popular block cipher cryptographic primitive is Data Encryption Standard (DES) [11]. There are also widely used other block cipher algorithms such as Advanced Encryption Standard (AES) [12], RC5 [13] and Blowfish [14].

## 1.5 Public Key Cryptography

Public Key Cryptosystem (PKC) differs from Symmetric Key Cryptosystem according to key count. PKC uses two separate keys, one of them is the public key the second is the private key. The owner secretly keeps private key, whereas the owner or a trusted third party broadcast the public key. It is computationally infeasible to calculate private key by exploiting the public key.



Figure 8. Public Key Encryption [17]

PKC is used for confidentiality purposes, such as encryption and decryption. Also it is used for authorization purposes such as 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 functions for confidentiality purposes as shown in Figure 8.



Figure 9. Validating a Signature [18]

Authorization and verification purposes are met by using private key as the encryption key as depicted in Figure 9. Since no one could know the private key of the owner, only private key owner could produce the encryption of a plaintext encrypted with a private key. This kind of encryptions could be decrypted using the public key. Since the public keys are broadcasted, anyone could verify the digitally signed plaintext. Therefore usage of private keys in encryption does not meet the confidentiality purposes but only authorization purposes.

Digital signature mechanism consists of two parts. The first part is the *Signing* part. The sender processes a plain text with a signature algorithm using the private key. Signature algorithm produces a digital signature. Digital signature does not reveal the plain text unless it is subjected to a validation algorithm that uses the corresponding public key as parameter. The second part is *Verification* part. The receiver processes the digital signature with a validation algorithm by using the public key. Validation algorithm determines if the processed signature is valid.

Some of the widely known, well-regarded asymmetric key cryptographic algorithms are Diffie-Hellman Key Exchange Protocol [19], Digital Signature Algorithm (DSA) [20], ElGamal [21] and the most known one is RSA [22] algorithm.

# Requirements for a secure and seamless mIcropayment scheme In wIreless mesh networks

For a payment scheme designed for Wireless Mesh Networks requires following attributes:

* **Wide Coverage:** Users should be getting service within a large area.
* **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.
* **Seamless Connection:** Users should be able to switch between access points as they move without noticing it.
* **Seamless Roaming:** Users should be able to switch between operators as they move without noticing it.
* **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. Moreover, man-in-the-middle and replay attacks must be prevented.
* **Two-way honesty:** clients cannot deny that they did not take service. Operators cannot claim that they provide service more than they actually provided. These are to be guaranteed using strong cryptographic protocols.
* **Preventing Double Spending:** A payment token should not be able to be used to get more services that its value. In particular, the payment token should not be used twice or more.
* **Unlinkability:** 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.

## 4.1 Requirements of the Network

Secure and seamless pre-payment system for Wireless Mesh Networks will not only consists of mesh backbone but also Wi-Fi clients and wired servers. Mesh backbone will basically relay the packages from clients to server to make the users 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 802.11s protocol. Clients will use 802.11a/b/g Wi-Fi protocols to connect to the access points/mesh routers.

## 4.2 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. The basics of the system model, roles, entities and requirements have been identified in Deliverable 1. As mentioned there, our system model assumes 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 is also a TTP (Trusted Third Party), which may be reachable through operator. These system components are listed; together their icons used in the protocol figures, in Table 4.1.

Table 4.1. System Entities

|  |  |
| --- | --- |
| D:\My Documents\albert\tt proje\phone.png | Mobile user (client) |
| D:\My Documents\albert\tt proje\accessPoint.png | 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. |
| D:\My Documents\albert\tt proje\cloudWithoutDots.png | Mesh backbone of the operator |
| D:\My Documents\albert\tt proje\gateway.png | Gateway (GW) that connects the mesh backbone to outer world and also to the operator's server |
| D:\My Documents\albert\tt proje\operator.png | Operator's server (OP). Keeps necessary logs and user info. |
| D:\My Documents\albert\tt proje\trustedThirdParty.png | Trusted Third Party (TTP). Payment related logs are mostly to be generated by the TTP. |

Since the clients are mobile, they may hand over 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 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 unlinkability 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 protocols detailed in this deliverable are designed by considering the abovementioned requirements. The symbols used in this document are given in Table 4.2.

Table 4.2. The list of the symbols

|  |  |
| --- | --- |
|  | XOR operation |
|  | Concatenation |
|  | Encryption of using the key |
|  | Decryption of using the key |
|  | Taking hash of times |
|  | Taking HMAC of using the key |
|  | th element of the hash chain (usage order) |
|  | Public key of TTP |
|  | Private key of TTP |
|  | th Access Point or its identity |
|  | th Operator or its identity |
|  | Public key of |
|  | Private key of |
|  | Serial Number |
|  | Nonce created by entity |
|  | Previous Alias |
|  | New Alias |
|  | Public key certificate of |
|  | Initialization Vector |
|  | Timestamp |
|  | Connection Request |
|  | Disconnection Request |
|  | Roaming Request |
|  | Change Alias Request |
|  | Mobility Request |
|  | Response (used in various protocol as positive acknowledgment) |
|  | Disconnection Acknowledgement |
|  | Roaming Acknowledgement |
|  | Mobility Response |

## 4.3 Connection Card Structure

*Connection Card* is the main deed that clients buy from operators and use to get Internet service. We use a prepaid system, in which connection cards include credits as tokens. 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 for getting Internet service are basically links in a hash chain. For each set of tokens, the operator picks on a random *Initialization Vector* (*IV*) 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 IV is taken hundred times. More formally a hash chain with, say 100 tokens are constructed in the following way.

…

is the first token to use. Then we use the token in the increasing order of token index. In this way, we exploit one-way property of hash algorithms such that an attacker cannot learn the next token even if she knows the previous ones.

The operators inform the TTP (Trusted Third Party) about the associations between and corresponding so that TTP validates them as needed in the protocols.

Connection Cards are refillable with hash tokens, which are to be sold by the operators. We assume a free market strategy in the marketing of the hash tokens. The prices or campaigns related for the marketing of hash tokens are to be decided by the operators. In other words, operators would compete with each other to sell hash tokens. They also 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 is a 16 digit alphanumeric and case sensitive value. With this setting, the system is able to support up to users. Hash tokens are to be generated using SHA-256 hash algorithm; hence they are 32 bytes long.

Considering current technology, smart cards are suitable tools to be connection cards. A simple Connection Card with 4 KB memory could store a and more than 100 hash tokens.

## 4.4 Alias Computation

The clients employ aliases as temporary identifiers. Changing aliases in a regular basis provides anonymity to some extent. The proposed system practices a mechanism in which the aliases could be computed by the clients and also by the TTP.

Serial number () of the connection card is used as the base for client’s aliases. The computation process of an alias is given below:

1. Client picks a random 128-bit unsigned number and calls it his nonce, .
2. Perform XOR operation with and this nonce, the resulting value is the alias:
3. Client uses this alias whenever his identity is required.

Aliases are to be updated periodically. The related protocols will be explained later in this document, but in this section we explain how a new alias is computed using the old previous one. Basically, the client picks a new nonce and XORs this nonce with its current alias to compute the next one. More formally, in order to change previous alias, , to a new alias, , the client performs the following steps.

1. Client picks a random 128-bit unsigned number as a nonce value, .
2. Perform XOR operation with and this nonce, the resulting value is the new alias:

The nonce values used in substitution 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.

One may argue that this kind of alias computation would have a risk of causing same alias for several users. 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. The problem is addressed by making TTP to check proposed alias to be a unique alias at that point of time. This check is embedded in related protocol, which will be described later.

# Protocols of the secure and seamless mIcropayment scheme

## 5.1 Initial Authorization

Initial Authorization is the beginning for system usage. Whenever a client purchases some new hash tokens from an operator, she will need to authorize herself to TTP. Initial Authorization Protocol, shown in Figure 1, achieves mutual authentication and authorization of the user.

In Figure 1, connection between client and serving access point (APs) is Wi-Fi (802.11b/g). 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.



Connection between serving access point and relaying access point is wireless, and uses 802.11s protocol [5]. The mesh backbone is like a cloud from the mobile user’s perspective. It is a black box; which receives packets from the client and delivers them to the gateway in a multi-hop manner. Mesh backbone uses Hybrid Wireless Mesh Protocol (HWMP) [6], which is a hybrid routing protocol. Once the mesh nodes deliver apacket through a route, the path between source and destination is stored in tables of serving access point. Future deliveries use the stored route in the table and decrease packet delivery delay.

Connection medium between mesh backbone and gateway () is wireless. Gateways and operators communicate through secured wired medium. Operators and TTP communicate through secured wired standard using 802.3.

Mobile clients introduce themselves to the operator using *Initial Authorization* protocol. TTP already knows mobile user’s serial number () and first element of her hash chain (). The mobile user does not want to reveal her to any adversary because that will be used continually; it is as valuable as mobile client’s identity. To achieve anonymity, the mobile client computes an alias and uses this value instead of . The mobile client will change her alias periodically as she continues to get service (related protocols will be explained later).

Initial Authorization steps are described below.

1. Client computes an alias using a nonce that she generated.
2. Client forms a connection request and encrypts this connection request using TTP’s public key, with RSA-2048.
3. (Assume that connection card has 100 credits)
4. Client sends this to .
5. receives the connection request and relays the request through mesh backbone.
6. Gateway receives the *CR*and relays it to the operator.
7. Operator relays *CR*to *TTP*.
8. TTP receives connection request () and decrypts it using its private key.
9. TTP checks alias' uniqueness within its database of users, it would make the client start over the protocol if alias is not unique.
10. It computes
11. TTP checks and association. Store and
12. TTP computes
13. TTP sends to theOperator.
14. Operator receives and verifies the signature using public key of TTP.
15. The Operator gets and and stores these values. The value of is the client's alias until she changes it.
16. Operator sends to the gateway.
17. receives and verifies the signature using public key of TTP.
18. It stores and .
19. forwards to through mesh backbone.
20. receives and verifies the signature using public key of TTP.
21. It calculates and and stores these values.
22. forwards to the client.
23. Client obtains and comprehends that she is authorized to get service.

The packets have been sent in encrypted manner in hop-by-hop basis during their route. Every access point knows it’s neighbors’ public keys. For the sake of simplicity, encryption and corresponding decryptions are not shown in the figure.

## 5.2 Reuse of a Connection Card

The clients may disconnect before using up all the credits in a connection card. *Reuse of a Connection Card* protocol allows the clients to connect using the remaining credits in a card. This protocol does not differ widely from *Initial Authorization* protocol. The main difference here is instead of sending first hash token; the client sends whichever token is the next one. Alias will change before the protocol starts. In the example depicted in Figure 2 client sends *i*th hash token and try to authenticate herself again. Another difference is that the client changes its alias during the connection. 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.

Figure 2 demonstrates how the protocol actually works end-to-end.



*Reuse of Connection Card* protocol is described below.

1. Client computes a new using a nonce that she generated.

*,* Where stands for *Previous Alias* of the client

Client forms a connection request and encrypts this connection request using TTP’s public key, with RSA-2048.

Client sends this to .

1. receives this connection request and relays it to the through the mesh backbone.
2. Gateway relays to the operator.
3. Operator relays to TTP.
4. TTP receives connection request () and decrypt it using its private key.

TTP checks alias' uniqueness, start over the protocol if necessary.

It computes

It checks and association and stores (i.e. the new alias) and .

It computes

It sends to operator.

1. Operator receives and verifies the signature using public key of TTP.

It getsand , stores these values.

Operator sends to the gateway.

1. receives and verifies the signature using public key of TTP.

It gets and and store these values.

sends to the .

1. receives and verifies the signature using public key of TTP.

It gets and and store these values.

sends to the client.

1. Client gets and understands that she is authenticated to get service.

In this protocol, we assume that the packet has been sent in encrypted manner in hop-by-hop basis during its route. For the sake of simplicity, we have not shown this encryption and corresponding decryptions in the figure.

## 5.3 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.

Figure 3 describes the protocol briefly.



APS authentication is described below.

1. sends a challenge request to the client which started connection.
2. When client receives this challenge request, it sends a 128 bit challenge to
   1. Client drops the packet if it is not the that she sent connection request.
   2. Client drops the packet if there was not any authentication request.
3. hashes this challenge, and uses relevant hash value (here , but it could be any if the authentication protocol runs after the reuse of a connection card protocol) as the key of :

sends response to the client.

1. Client also HMACs the challenge and uses the stored hashvalue () 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.

## 5.4 Packet Transfer

After mutual authentication of client and client starts to send packets as shown in Figure 4.



1. Before sending data packet, client sends next token first. (It causes client to spend one token for each connection session.)
2. gets from the database. Remember stored client’s alias and hash token in *Initial Authorization* and *Reuse of a Connection Card* protocols.

Then:

* 1. Checks if
  2. If true sends acknowledgement () toClient and updates currently used hash value as .

1. Client sends first 1024 byte data packet .
2. serves the client without wanting any other hash token for a predefined value. This value depends on the operator.
   1. Every time client sends data packets access point will update the amount of service the user got.
   2. Whenever service amount value passes the predefined threshold client will send the next hash token.

## 5.5 Changing Alias

We achieve anonymity property by using aliases, but tricky part here is achieving unlinkability. We have to change aliases on a basis that an adversary, who knows a certain client’s alias, could not be able to trace client’s activity on his home network, and also could not trace his movements among the operators or access points.

To be able to change alias in a safe way, we have to communicate with TTP but we do not want to bother TTP very often because that would slow down the entire operation due to extra delays caused. That’s why we make use of periodic timer value to make sure that access points would ask 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 [7].

In this way, we prevent any type of attack that would aim to analyze network traffic of access points and examine connection requests. By passing the task of enforcing alias changing to the access points, we gain a more generalized control over the clients. No attacker would understand which client wanted to change his alias, because all the clients getting service from a particular access point have requested to change their aliases at that particular time.

We need to keep actual change alias operation on the client, because client and the TTP should be the only ones who know association between an alias and a client’s .



Changing Alias Protocol is shown in Figure 5 and described below.

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

When the Alias Change Timer value expires, Access Point broadcasts "Change Alias" command to all of its clients. This times value is a system parameter; typical value is a couple of hours.

1. Client receives "Change Alias" command.

Client computes a new alias by picking up a new random nonce and computing , where is the new alias and is the previous alias.

Client forms a Change Alias Request ()

It sends this to .

1. receives and relays it to the via mesh backbone.
2. Gateway forwards to operator.
3. Operator relays to TTP.
4. TTP receives Change Alias Request () and decrypts it using its private key.

TTP checks for new alias' () uniqueness and starts over the protocol if not unique.

TTP computes

It checks and association and stores and.

It computes (

TTP sends to operator.

1. Operator receives and verifies the signature using public key of TTP.

It gets and , and stores these values.

Operator sends to the gateway.

1. receives and verifies the signature using public key of TTP.

It gets and , and stores these values.

sends to the .

1. receives and verifies the signature using public key of TTP.

It gets and , and stores these values.

sends to the client.

1. Client gets the .

Client decrypts it using TTP’s public key and update his last used hash value and new alias.

In this protocol, we assume that the packet has been sent in encrypted manner in hop-by-hop basis during its route. For the sake of simplicity, we have not shown this encryption and corresponding decryptions in the figure.

## 5.6 Update Packets

In our usual flow, after authentication, access points do the accounting. Because of the fact that they 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. But it is essential to send periodic updates to the operator. This is essential because we want to provide a seamless mobile communication, even when user steps out from one access point’s region to another’s. In this kind of situation, clients should authenticate themselves by showing themselves to gateway only. By doing that, we bypass operator and we can decrease authentication time significantly.

In order to use abovementioned protocol, gateways should be aware of client’s lastly used token and connection status. From security point of view, it would be ideal to update gateway entry at every time when the client uses a new token. However, this would be very inefficient and would increase network traffic. That’s why we define threshold time values for access points and gateways. After passing this threshold time values, access points send update packets to gateways, and gateways send update packets to operator. This mechanism is depicted in Figure 6 and explained below.



1. After client sends the first token she uses, at the current session, access point starts to count the time passed. After units of time (value of is a system parameter), access point sends lastly used hash token to the relaying access points.
2. Relaying access points forward the token to the gateway.
3. Gateway receives the token and updates the client entry. Gateway updates the last used value for the token.
4. Gateway starts to count the time passed from the lastly arrived token. After units of time (value of is a system parameter), gateway sends lastly used token to the operator.
5. Operator receives the token from gateway and updates the client entry by changing the last used value attributes with the newly received token.

## 5.7 Disconnection

To be able to run Reuse Connection Card Protocol, we have to establish proper disconnection. Our Update Packets protocol brings stability to the system in case of a connection interruption, but we assume that most of the users will be disconnecting from the operator using the disconnection protocol that we explain in this section and in Figure 7.

These protocols are designed for the sake of operators, to make them aware of how many users they are serving at a point of time. That information will bring them the opportunity to organize their servers accordingly, deciding on their marketing strategies using traffic density, etc.



Disconnection protocol is shown in Figure 7 and described below.

1. Client forms a disconnection request

Client sends it to the access point.

1. Access Point receives and prepares itself to disconnect that particular client.

AP relays to the mesh backbone, to make it reach to the .

1. receives and forwards the to the Operator.
2. Operator receives and forwards the to the TTP.
3. TTP receives the and . It checks the association between the and this hash token; if the association holds, then it computes a disconnection acknowledgement ().

TTP sends to Operator.

1. Operator receives , verifies the signature on it and marks client as disconnected.

Operator relays to .

1. receives, verifies the signature on it and marks client as disconnected.

It relays to the mesh backbone.

1. Serving Access Point eventually gets the , verifies the signature on it and disconnects the particular client, which corresponds to the it received.

In this protocol, we assume that the packet has been sent in encrypted manner in hop-by-hop basis during its route. For the sake of simplicity, we have not shown this encryption and corresponding decryptions in the figure.

## 5.8 Distributing Access Point Public Keys

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

In Figure 8, 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.



Distributing 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 (); 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.

1. TTP receives the and starts to generate certificates for every access point and public key pair.

Certificates are formed as:

TTP stores these certificates for distribution.

We make use of other protocols (such as *Initial Authorization* or *Reuse of a Connection Card* 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 together to operator.

1. Operator receives the connection response and the certificate and relays these packets to the access point through gateway and mesh backbone.
2. Access point gets its certificate and broadcasts it to the nearby access points. It also stores this certificate.

## 5.9 Seamless Roaming (Payment Related)

When the clients need to get service from an access point of a new operator, they roam between old operator and new one. In this kind of situations, we do not bother TTP and save time and computational power.

Because of the fact that every access point has its public/private key pairs and ability to broadcast public keys, we can handle roaming in a seamless way without running the authorization process from scratch. As it is shown in Figure 9, 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 operator.



Roaming protocol is shown in Figure 9 and described below. In this protocol, the client would like to switch from its old operator () to a new one (). In this setting, is the last access point that the client got services from . , is the access point that the client would like to continue to get services in network.

1. Client sends a Roaming Request () to .
2. receives and forms a Roaming Acknowledgement ().

sends to the client.

includes the roaming ticket that the client uses to get services from . It is signed by and encrypted for , thanks to public key distribution mechanism that we employ.

1. starts the disconnection protocol for the client after sending .

This disconnection protocol runs in parallel with the roaming protocol. Thus it does not put an extra delay in roaming. Old operator () stores disconnection acknowledgement () 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 in order not to get confused when disconnects without a connection request reached to it.

In this scheme, ’s signed ticket serves as a formal document, which represents the beginning of the session with .

1. Client receives and forwards it to the new operator ().
2. decrypts using its private key.

It gets the signed ticket of the . stores this signed data to use it for collecting funds from TTP.

verifies the signature over this signed ticket using ’s public key. Then, it checks in order to decide whether the ticket has expired or not.

Then, starts a challenge-response protocol with the client.

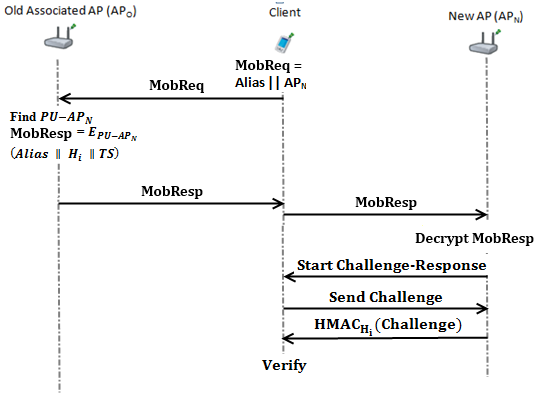
1. Client sends a 128-bit challenge to .
2. computes

sends this HMAC value to the client.

1. Client receives this HMAC. Moreover, it computes the same HMAC using . If the computed one and the received one are the same, then it authenticates .

## 5.10 Seamless Mobility in Home Operator (Payment Related)

When a client moves out of the coverage area of its associated AP or if another AP provides a better service, the client may want to hand off to another AP. For the payment purposes, we develop a seamless mobility protocol in order to avoid a full authorization process during such a handoff. This protocol is shown in Figure 10. The user sends only the ID of the new access point. This protocol uses the service of “Distributing Access Point Public Keys” protocol. With the ID of the new access point, the old access point pops up the public key, and uses this public key to encrypt the credentials of the user. In this way, the necessary user and payment information to continue the service at the new AP is sent to it in an encrypted way.



In Figure 10, client moves to the new AP’s () coverage area from old AP (). Seamless mobility is described as follows:

1. Client forms a mobility request () and sends it to .
2. concatenates Alias, , timestamp , and encrypts it with the public key of . It relays this mobility response () to via client.
3. decrypts using the private key of itself and checks whether it is expired or not.
4. If it is not expired, stores and *.* Then sends a challenge request to the client.
5. When client receives this challenge request, it sends a 125-bit challenge to .
6. computes

sends this HMAC value to the client.

Client receives this HMAC. Moreover, it computes the same HMAC using . If the computed one and the received one are the same, then it authenticates.

# Payment to the operators (settlement)

In our system, operators claim their money from the TTP, as they show their service logs. A log proves a service that has been provided between a connection request and a disconnection request.

Operators store CRs of the clients; we described CRs in Initial Authorization and Reuse of a Connection Card protocols. When a client makes a disconnection request, operator stores the disconnection request as well. After receiving the DR, operator forms its log as follows.

Where TS stands for timestamp. We make operators to add timestamps to make TTP’s job easier.

Let’s look at what is going to happen when TTP receives these two logs from an operator.

1. TTP will sort the logs according to their TS value; this sorting process would ease the operation.
2. TTP first processes CR. CR is a request; which 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

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

Note that SN’s first token used is Hf.

1. 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 it is a valid log.
2. 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; that is why that 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 hash token from the decrypted DR. TTP makes the XOR operation: and gets the SN. Note that SN used is the hash token came with the DR to end the service.

1. 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.
2. 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.

We need to consider misuse of logs by the operators. Consider the situation of a client:

* Gets service from his home operator between H0 and H10
* Gets service from a foreign operator between H11 and H20
* Gets service from his home operator between H21 and H30

In this type of situation home operator has two CRs and DRs, whereas foreign operator has a CR and DR. Let’s look at the look at the logs of home operator:

We see that home operator served between H0 and H10 also served between H21 and H30. Home operator would want to take money for serving between H11 and H20. It could pretend that it has served the client between H11 and H20 by not sending Log2 and Log3. Because Log2 indicates that client is disconnected from the operator at H10 and Log3 indicates that Client started to get service from the operator at H21. By sending only Log1 and Log4 home operator tells TTP that it served the client between H0 and H30. Operator would want money for serving 30 hash tokens.

In that kind of situations there should be a foreign operator, which has served between H11 and H20. Foreign operator would have two logs as follows.

Foreign operator proves that it has served between H11 and H20 by showing the signed RP and DA.

TTP would see that it has already paid home operator for service to that particular client between H11 and H20. This means that home operator tricked TTP to pay more.

In our system TTP is the one who has the authority, it pays operators their money. If it 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.

# discussion

In this section, we discuss which of the requirements are met.

Roaming/mobility: Reuse of a connection card is possible after attempting first connection. Roaming is supported, when our protocol is implemented in participating *AP*s, and tokens are valid.

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

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

Anonymity: For legal purposes users must give their identities to connection card issuer (*TTP*) for getting connection cards. Therefore, as far as *TTP* keeps clients’ identities secret, users can stay anonymous.

Mutual authentication: We have seen how the server authenticates the client. The AP receives valid token information, and with the challenge-response protocol both AP and the Client is mutually authenticated.

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

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

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

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

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

Unlinkability: Our protocol provides unlinkability by changing aliases periodically. There occurs linkability between the times clients change their aliases. The period of time to change the aliases is a choice of the designer.

# Unit test results

Unit tests cover protocol behaviours under low pressure. In these tests there is only one user, and this user performs the same protocol every minute. These tests are done to ensure that modules of the system are fit for use.

As discussed earlier 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.

## 6.1 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. Charts present 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 8 gives the result for unit test of end-to-end two-way protocols.



Figure 8. End-to-End Two-Way protocols Unit Test Result

As shown in Figure 8, there is a delay that shows variation around 0.04 second. This unstable behaviour is caused by different initial packet delays. System needs some packets to set up paths between mesh nodes. The performance stabilizes in time. Average delay shows a peak by the end however the difference between highest and lowest values of the results is inconsiderable.

## 6.2 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 9, shows the average delay of the protocol versus time.



Figure 9. Access Point Authentication protocol Unit Test Result

As shown in Figure 9, 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.

## 6.3 Unit Test Result for Seamless Mobility and Roaming

*Seamless Mobility* and *Seamless Roaming* protocols have the same behaviour 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 consists 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 10 presents the unit test result for *Seamless Mobility* and Roaming protocols.



Figure 10. Seamless Mobility and Roaming protocols Unit Test Result

In unit test for these protocols, a 0.15 second of network delay for access point change is observed. Similar to other protocols, there is a transitive period at the beginning of the simulations, however it reaches steady state in time and gains balance.

## 6.4 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 it’s often use. Unit test scenario of *Packet Transfer* protocol is that a client sends a 512-byte packet every minute.

Figure 11 shows the unit test result for Packet Transfer protocol.



Figure 11. 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. As seen on Figure 11, at steady state, packets are received in a very short amount of time, which is around 0.0002 second.

## 6.5 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 12 shows the average delay of *Update Packets* protocol over time.



Figure 12. Update Packets protocol Unit Test Result

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

# User Modeling And Mobility

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 [8]. Roaming and handoff related actions are triggered by user mobility.

## 7.1 User Actions

In real-life scenario simulations, network usage related actions are modelled using two-state Markov Chain as shown in Figure 13. 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 13. State Diagram of Clients

The initial state is *Not Connected*. In this state, the user switches to *Connected* state with the probability value of . 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 .

While in *Connected* state, the user remains connected (i.e. stay in the same state) with the probability of . 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. This transition disconnects the user via *Disconnection* protocol.

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

(1)

Where, denotes .

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

(2)

Where, denotes .

## 7.2 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 modelled in three time slots: (i) night (00:00 – 07:59), (ii) daytime (08:00 – 15:59), and (iii) evening (16:00 – 23:59).

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 night 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 *night* 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 nights. 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 and are determined based on the abovementioned discussion about the client type characteristics and the time slots. These values are given below. The triplet specify the probability values for night, daytime and evening, respectively.

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 minutes between connections. Once connected, average connection time for this category is minutes.

## 7.3 User Mobility and Timing

Real-time scenario covers Internet usage of 300 users in a 1-km2 metropolitan area. The simulations time begins at 00:00 a.m. and lasts for 24 hours. Simulation time is divided into 3 parts considering night, daytime and evening. Every part of the day has different statistical values for client behaviours.

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 14. 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 closest to client’s current location.



Figure 14. User Mobility

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 between2 km/h to 6 km/h. The clients are assumed to move without a motor vehicle.

# Results for Real-Life Scenario Simulation

Results for unit test simulations are described before; however the most significant results are real-life scenario simulation results. Despite the randomness of the system, users’ actions are highly related to their group and current simulation time.

Charts for the results display the average delay for a particular protocol.

## 8.1 Overview

Final simulations provided the results in Table 2. Charts on Figure 15 and Figure 16 are drawn exploiting the results in Table 2. Considering the results it could be calculated that over 100 minutes of Internet service, workers have only waited for 1 minute for system delays. In average, over 1000 minutes of Internet service needs a delay of 13 to 16 minutes of waiting.

TABLE II  
Simulation Results for Client Types

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Total Internet Usage Time | Total Internet Usage Delay | Average Internet Usage Time for a Client | Average Internet Usage Delay for a Client |
| Student | 95899,26 Minutes | 1698,95 Minutes | 958,99 Minutes | 16,98 Minutes |
| Worker | 101681,64  Minutes | 1316,35 Minutes | 1016,81  Minutes | 13,16 Minutes |
| Non-Worker | 105335,08 Minutes | 1456,12 Minutes | 1053,35 Minutes | 14,56 Minutes |



Figure 15. Total Amount of Internet Usage Times for Client Types vs. Total Delays



Figure 16. Average Usage Times for Client Types vs. Average Delays

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

Figure 15 and Figure 16 shows the overall results for real-life scenario simulation. Figure 15 shows comparison of minutes clients used as idle or active. Figure 16 shows the average value for the clients of the same group.

## 8.2 Real-Life Scenario Simulation Result for Initial Authorization



Figure 17. Result for Initial Authorization Protocol

*Initial Authorization* protocol is used at the beginning of the service for each user. As it is seen on the chart every one of the 300 users are authenticated at the end of 40th minute.

Simulation starts around the 10th minute in the morning. At the beginning there is a huge amount of users, trying to authenticate. Figure 17 indicates that, this process varies between 0.6 and 2.5 seconds. After 10 minutes it attains a balance and *Initial Authorization* protocol meets a delay of 1 second, which means when users open up their mobile device they would have Internet service after 1 second.

## 8.3 Real-Life Scenario Simulation Result for Reuse of a Connection Card Protocol



Figure 18. Result for Reuse of a Connection Card

*Reuse of a Connection Card* protocol is used after disconnecting from the system. As it is seen it is a highly used protocol in the system. It starts around the 50th minute and used for the entire time of the simulation.

As seen on Figure 18, at the beginning of the protocol the delay changes between 0.1 and 0.6 second. After some time protocol achieves a balance and a 0.4 second of network delay is observed.

## 8.4 Real-Life Scenario Simulation Result for Changing Alias



Figure 19. Result for Changing Alias Protocol

Every active client uses *Changing Alias* protocol in the system in every 50 minutes. The protocol is first used at 50th minute and it is used entire time of the simulation.

As one can see on Figure 19, at the beginning of the protocol the delay for the protocol varies between 0.1 and 0.4 seconds. After some time the average delay for the protocol converges to 0.4 seconds.

## 8.5 Real-Life Scenario Simulation Result for Disconnection



Figure 20. Result for Disconnection Protocol

*Disconnection* protocol first appears around 30th minute and it is used through the entire time of the simulation. Figure 20 shows that, at the beginning of the system Disconnection protocol average delay vary between 0.1 and 0.5 second but through time the average delay meets 0.4 second.

## 8.6 Real-Life Scenario Simulation Result for Update Packets



Figure 21. Result for Update Packets Protocol

*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 10 minutes.

As it is seen on Figure 21, at the early stages of the protocol, the average delay value varies between 0.6 and 1.4 second but then after some time the protocol stabilized around 0.4 second.

## 8.7 Real-Life Scenario Simulation Result for Seamless Mobility in Home Operator Protocol



Figure 22. Result for Seamless Mobility in Home Operator Protocol

*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.

By looking at Figure 22, it could be said that*, Seamless Mobility* protocol has an initial average delay that shows difference between 0.2 and 1.2 seconds. A user loses around 0.1 second to make a handover to the new access point.

## 8.8 Real-Life Scenario Simulation Result for Roaming Protocol



Figure 23. Result for Roaming 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.

*Roaming* protocol has an average delay that varies between 0.05 and 0.2 seconds. There are 2 operators so a client has a %50 chances to make a *Seamless Mobility* or *Roaming* protocols. After some time protocol reaches a balance around 0.2 second of delay.

As one can see on Figure 23, the results for *Roaming* protocol shows a boost until the middle of the simulation time but it decreases and achieves a balance

## 8.9 Real-Life Scenario Simulation Result for Packet Transfer



Figure 24. Result for Packet Transfer Protocol

*Packet Transfer* protocol is the mostly used protocol in the system.

Figure 24 states that, at the beginning of the protocol the average delay value varies between 0.005 and 0.025 but then the protocol achieves a balance around 0.02 second.

# Conclusion

In unit tests, standalone performances of the protocols under trivial usage scenarios are analysed. Unit tests set an example for how the system will behave in empty hours. In this way, the first proof-of-concept implementation of the system is provided and showed that the designed protocols reach steady state.

Uniform probability distribution model enables us to simulate real time scenarios in simulation environment, and gets results closer to real time situations.

There exist different user types, as there are different types of clients in real life. There is also randomness in the system, so there could be different outcomes for the same simulations. The simulations implemented to cover even the most unexpected situations.

Results are significant since the actual usage of the system is a combination of these protocols. Unit tests and real-life scenario simulation results show that the proposed system is a considerable and an effective pre-payment system.

# References

1. Akyildiz, I. F., Wang, X., and Wang, W. (2005) Wireless mesh networks: a survey, *Computer Networks and ISDN Systems*, 47(4): 445-487.
2. Intel Inc., Multi-Hop Mesh Networks—a new kind of Wi-Fi network.
3. J. Walker, Wi-Fi mesh networks, the path to mobile ad-hoc. Available from [http://www.wi-fitechnology.com/ Wi-Fi\_Reports\_and\_Papers/Mesh\_Networks\_References.html](http://www.wi-fitechnology.com/%20Wi-Fi_Reports_and_Papers/Mesh_Networks_References.html).
4. Vaughan-Nichols S.J., (2004) Achieving wireless broadband with WiMax, *IEEE* *Computer*, vol. 37, no.6, pp. 10-13.
5. The ZigBee Alliance. Available from: <http://www.zigbee.org>.
6. Network Simulator 3 (ns-3) <http://www.nsnam.org>
7. Trappe, W. Washington, L. C. Introduction to Cryptography with Coding Theory, Second Edition, pp. 218-220
8. Oligeri, G. Chessa, S. Giunta, G. Loss Tolerant Video Streaming Authentication in Heterogeneous Wireless Networks, Computer Communications, 34(11): 1307-1315, 2011.
9. American Bankers Association, *Keyed Hash Message Authentication Code*, ANSI X9.71, Washington, D.C., 2000
10. H. Krawczyk, M. Bellare, and R. Canetti, *HMAC: Keyed-Hashing for Message Authentication*, Internet Engineering Task Force, Request for Comments (RFC) 2104, and February 1997.
11. FIPS PUB 46-3 (1999) Data Encryption Standard (DES), <http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf>
12. FIPS PUB 197 (2001) Announcing the Advanced Encryption Standard (AES), <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
13. Biryukov A. and Kushilevitz E. (1998). Improved Cryptanalysis of RC5. *EUROCRYPT* 1998.
14. Bruce Schneier (1993) Description of a New Variable –Length Key, 64 bit Block Cipher (Blowfish), <http://www.schneier.com/paper-blowfish-fse.html>
15. Robshaw, M. J. B. (1995) Stream ciphers, RSA Laboratories Technical Report.
16. W. Stallings, Cryptography and Network Security: Principles and Practices, 3rd edition, Prentice Hall, NJ, 2003.
17. Managed File Transfer and Network Solutions, <http://www.jscape.com/blog/bid/82975/Which-Works-Best-for-Encrypted-File-Transfers-RSA-or-DSA>
18. Coleridge, R. (1996) The Cryptography API, or How To Keep A Secret, *Microsoft Security Technical Articles*, <http://msdn.microsoft.com/en-us/library/ms867086.aspx>
19. RFC 2631—Diffie-Hellman Key Agreement Method E. Rescorla June 1999.
20. FIPS PUB 186-3 (1994) Digital Signature Standard (DSS) - CSRC, http://csrc.nist.gov/publications/fips/fips186-3/fips\_186-3.pdf
21. Elgamal, T. (1985) A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, *IEEE Transactions on Information Theory*, <http://caislab.kaist.ac.kr/lecture/2010/spring/cs548/basic/B02.pdf>
22. Rivest, R., Shamir, A., and Adleman, L. (1978) A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*,21(2): 120–126.
23. FIPS 180-3 (2008) Secure Hash Standard (SHS), <http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf>
24. RFC 1321 (1992) The MD5 Message Digest Algorithm, <http://tools.ietf.org/pdf/rfc1321.pdf>