Authors: Albert Levi and Can Serhat Leloğlu

Contents

1 Summary 2

2 Introduction and Background 2

3 Connection Card Structure 5

4 Anonymized Subhash Chains 6

5 Alias Computation 9

6 Designed Protocols 9

6.1 Initial Authorization and Reuse of a Connection Card 10

7 Payment to the Operators (Settlement) 13

8 Conclusions 15

9 References 16

# Summary

In our SSPayWMN project, our goal is to design a secure and fair payment scheme which also takes users' privacy and untraceability into account. Seamless handover and roaming among the operators in the above secure and fair setting will also be provided in SSPayWMN. In this document, we provide the design of the cryptographic protocols to be used to achieve our security goals under the light of the system and user requirements previously described in Deliverable 1.

With this document, we successfully complete work package 3 and first year work in the project in accordance with the statement of work. The implementation and the performance evaluation of SSPayWMN will be performed in a network simulator environment in the next work package (work package 4) which will be studied in the second year of the project, as planned in the statement of work.

# Introduction and Background

Wireless Mesh Networks [1] (WMN) is a multi-hop wireless networking technology to provide broadband ubiquitous access in metropolitan area. WMN provides flexibility for topology design and has self-organized nodes. These nodes could form routing tables and provide high-speed connection end to end. WMNs are easy to set up and they have manageable overheads.

Since WMNs are service providing system, they should have support for user identification, authentication as well as authorization and accounting. In our SSPayWMN project, our main goal 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 1.

Table 1. The system entities

|  |  |
| --- | --- |
| C:\Users\Public\Pictures\client.png | Mobile user (client) |
| C:\Users\Public\Pictures\ap.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. |
| C:\Users\SUUSER\Documents\GitHub\worddoc\thesisImages\meshBackbone.png | Mesh backbone of the operator |
| C:\Users\Public\Pictures\gateway.png | Gateway (GW) that connects the mesh backbone to outer world and also to the operator's server |
| C:\Users\Public\Pictures\operator.png | Operator's server (OP). Keeps necessary logs and user info. |
| C:\Users\SUUSER\Documents\GitHub\worddoc\thesisImages\ttp.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 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 protocols detailed in this deliverable are designed by considering the abovementioned requirements. The symbols used in this document are given in Table 2.

Table 2. The symbols used in the protocol specification

|  |  |
| --- | --- |
|  | 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 |
|  | Length of Anonymized Subhash Chain |
|  | Hash Token Renewal Interval |
|  | Anonymized Subhash Chain |

In our protocols, we use relevant cryptographic primitives. For public key encryption and signature purposes, we use 2048 bit RSA [1]. For symmetric encryption and decryption, we use AES-128 [2]. SHA-256 [3, 4] algorithm is employed as hash function and to form hash chains. For Challenge-Response protocols we use HMAC [3, 4] algorithm.

# 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* (), 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 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 is taken hundred times. More formally a hash chain with 100 tokens is constructed in the following way.

…

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 is a 128-bit 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 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 and approximately 1000 hash tokens.

# Anonymized Subhash 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 () and Hash Token Renewal Interval () determine the Length of Anonymized Subhash Chains () as following:

In unit simulations and real-life scenario simulations we have used and .

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

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

When the client wants to use the hash token 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:

Client and the TTP calculate the elements of the anonymized subhash chain by performing hash operations on as follows:

A sample for the generation of the anonymized subhash chains is depicted in Figure 4.1.



Figure .1. A Sample for Generation of Anonymized Subhash Chains with *,* and

Before any authentication or change alias phase, the client sends the first hash token of the remaining hash chain (In Figure 4.1 the first hash token is H0). TTP knows the 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 times. TTP computes the corresponding anonymized hash token and takes the hash of the output times and calculates . These operations form up an anonymized subhash chain and the anonymized hash tokens are spent in reverse order as shown below.

…

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:

Here suppose and are on the same hash chain, for an adversary it is infeasible to find out and by using or . 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 and . Therefore, we can say that usage of anonymized subhash chains provides unlinkability among different sessions.

# 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 () 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 .
2. Perform XOR operation with and her nonce; take the hash of the output.
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 .

# Designed Protocols

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.

## 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 6.1, 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 (). 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.

Macintosh HD:Users:canleloglu:Desktop:worddoc:thesisImages:protocolsInDetail:newPdf:initAuthReuse.pdf

Figure .1. Initial Authorization and Reuse-CC

Mobile clients introduce themselves to the operator using *Initial Authorization* protocol. in *Initial Authorization* protocol, in *Reuse of a Connection Card* protocol. TTP already knows mobile user’s serial number () and the first element, , of her hash chain. The mobile user does not want to reveal her to any adversary because that 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 . 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 that she generated.
2. (The is assumed to have credits)
3. Client sends this to .
4. Client starts to generate an anonymized subhash chain as the network processes client’s .
5. receives the connection request and relays the request through mesh backbone.
6. Gateway receives the and relays it to the operator.
7. Operator relays to TTP.
8. TTP receives the 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 generates anonymized subhash chain by taking the hash of times. The output of the last hash operation gives the first token of the anonymized subhash chain, which is .
14. TTP computes
15. TTP sends to the Operator.
16. Operator receives and verifies the signature using public key of TTP.
17. Operator sends to the gateway.
18. The Operator gets and and stores these values.
19. GW receives and verifies the signature using public key of TTP.
20. GW uses the shared secret key with and calculates
21. GW sends to the .
22. GW verifies the signature of TTP.
23. GW stores and .
24. receives and decrypts it using the shared secret key with GW.
    1. verifies the signature using public key of TTP.
    2. It calculates and 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.

# Payment to the Operators (Settlement)

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.

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

stands for timestamp in the logs. 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 value.
2. TTP first decrypts since it is encrypted with the public key of TTP. consists of , and the first hash token to be used to get service.
3. Consider
4. TTP decrypts it using its private key, and gets SN by the XOR operation:
5. Note that SN’s first token used is .
6. and are formed by using the tokens from the original hash chains. Therefore, anonymized subhash tokens are not needed to be used in settlement phase.
7. 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.
8. The abovementioned log is only a service starter; operator needs to show service-ending log to claim its money from the TTP.
9. Service ending log naturally has a larger value; therefore this log comes later in the sorted list of logs.
10. TTP takes the ending log and decrypts using its private key.
11. TTP gets , and the hash token from the decrypted . TTP makes the XOR operation: and gets the . Note that used is the hash token came with the to end the service.
12. 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 . If the values match, TTP considers the log as a valid service-ending log.
13. 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 and
* Gets service from a foreign operator between and
* Gets service from her home operator between and

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

The home operator has served between and and also has served between and . Home operator would want to take the money for serving between and . It could pretend that it has served the client between and by not sending and . Since indicates that client is disconnected from the operator at and suggests that the client started to get service from the operator at . Sending only and results TTP to think that the home operator has served the client between and . This way operator would want money for serving 30 hash tokens.

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

Foreign operator proves that it has served between and by showing the signed and .

TTP would see that it has already paid home operator for service to that particular client between and . 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.

# Conclusions

In this deliverable, we designed the protocols to be used in the SSPayWMN protocol. With these protocols, the following security 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. The clients in the same operator can move from one *AP* to another by carrying necessary credentials in a secure way and without a new authorization and service interruptions.

*Seamless roaming*: Mobility of the clients from one operator’s zone to another is also supported without service interruptions and without needing to get authorized from scratch.

*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 client is authenticated. Valid token information is received by the *AP*, and with the challenge-response protocol both AP and the Client is mutually authenticated.

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

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

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

*Three-way honesty*: Since the tokens are issued by *TTP*, only the *TTP* and connection card holder knows all the tokens that are related with a specific connection card. Hence whenever a Client sends a new token, it is not possible for him to say “I did not use it”. Since *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 protocols provide unlinkability by changing aliases periodically. During the time of validity of a particular alias, the connection of a client can be traced, but once the alias has changed using the designed protocols, the client cannot be traced. The period of time to change the aliases is a system parameter.

With this document, we have finished the planned tasks of the first year of the SSPayWMN project. Next stage is the implementation of the protocols in the selected simulator (ns-3). We will also define the user profiles and parameters for the simulations. Finally, we will run the simulations and make the performance analyses. We expect to finish all these tasks and the project until the end of March 2013, as planned.

# References

1. 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.
2. FIPS PUB 197 (2001) Announcing the Advanced Encryption Standard (AES), http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
3. Trappe, W., and Washington, L. (2006) Introduction to cryptography with coding theory, *Person Education, Inc.*
4. Stallings, W. (2006) Cryptography and network security, *Person Education, Inc.*
5. Camp, J. D., and Knightly, E. W., (2008) The IEEE 802.11s Extended Service Set Mesh Networking Standard, *IEEE Communications Magazine*, August 2008, pp. 120 – 126.
6. Yang, K., Ma, J-F., Miao, Z-H., (2009) Hybrid routing protocol for wireless mesh network, In *Proceedings of Computational Intelligence and Security – CIS ’09*.
7. Chaum, D., (1982) Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1982.