# A SCTP-based Authentication Protocol: SCTPAP

Malek Rekik[1], Amel Meddeb-Makhlouf[2], Faouzi Zarai[1], Mohammad S. Obaidat[3] and K.F. Hsiao[4]
[1]*LETI laboratory,University of Sfax, Tunisia*
[3]*Computer Science and Software Engineering Department, Monmouth University, NJ 07764, USA*
[4]*Dep. of Information Management, Ming-Chuan University, Taoyuan County 333, Taiwan*

Keywords: Multihoming, *SCTP*, Security, Authentication, *AVISPA*, *SPAN*.

Abstract: Multihoming is among the features of *SCTP* (Stream Control Transmission Protocol), which makes it more robust and efficient than *TCP*(Transmission Control Protocol)but more vulnerable under attack. Nevertheless, a strong security can degrade the *QoS*(Quality of Service) by adding additional delay. Therefore, we propose in this paper, a secure authentication protocol that supports the establishment of multiple connections to protect multihoming networks with the least number of messages, number of parameters in each message and number of communicating nodes. The proposed scheme provides lower delay of authentication and protects against several attacks. Our devised protocol is analyzed using *SPAN* (Security Protocol Animator) for *AVISPA* (Automated Validation of Internet Security Protocols and Applications) tool. The obtained validation results show that the scheme is safe.

## 1 INTRODUCTION

Multihomed protocol is a mechanism that makes the host able to connect several networks under different IP (Internet Protocol) addresses by using different network interfaces. With traditional *TCP*, multiple connections are required to provide multihoming services, which involves the use of multiple ports. This makes the network management difficult and the communication during the change of address may be interrupted. Nevertheless, *SCTP* (Cano, 2011), which is a recent *IETF* transport layer protocol, supports multihoming. Indeed, it ties one connection called association in *SCTP* to several network interfaces at each communicating node. Transport addresses are exchanged during the initialization phase of an *SCTP* association. This phase consists of four-way handshake to protect against denial-of-service attacks. Even though, *SCTP* is more robust against network failures or congestion by dynamically selecting a path. Its features make it more vulnerable to the man-in-the-middle and hacking attacks. Among the related security solutions proposed by researchers are: *SCTP over IPsec* (Internet Protocol Security) (Bellovin et al., 2003), *SCTP-under-TLS* (Transport Layer Security) (Jungmaier et al., 2002), Secure SCTP (Hohendorf et al., 2006), and the extension AUTH-SCTP (Tuexen et al., 2007).

In this paper, we will introduce the proposed protocol, called secure optimized authentication for *SCTP* (*SCTPAP*) scheme. Our goal is to secure *SCTP* communication considering the following requirements in the design of *SCTP AP*:

- Integrity
- Confidentiality
- Mutual authentication
- Mutual belief on the session key
- Delay of authentication and re-authentication

We will use *AVISPA* tool for validation and security analysis of the proposed protocol.

The rest of this paper is organized as follows: Section II presents some existing security solutions, where their limitations are highlighted. Section III describes our *SCTPAP* scheme. Section IV contains the validation and analysis of the proposed algorithm using the *AVISPA* tool. Finally, conclusion and future works are provided in section V.

## 2 RELATED WORK

### 2.1 SCTP over IPsec

The feature of *SCTP* is not well supported by *IPsec*. The ref (Cano, 2011) identifies the problem of *SCTP*

Rekik M., Meddeb-Makhlouf A., Zarai F., Obaidat M. and Hsiao K..
A SCTP-based Authentication Protocol: SCTPAP.
DOI: 10.5220/0005123400470053
In *Proceedings of the 5th International Conference on Data Communication Networking* (DCNET-2014), pages 47-53
ISBN: 978-989-758-042-0

over *IPsec*: the *SCTP* sessions combine a group of senders at a group of receivers.

This has two impacts on the tunnel establishment procedure of *IPsec*, (Cano, 2011) where:

- The *SPD* must find a unique *SA* from a new type of triplet ({destination address group}, SPI (Security Parameters Index), *AH / ESP* (Authentication Headers / Encapsulating Security Payloads)). So, it is recommended that the *SPD* (Security Policy Database) entries are generalized in the form of groups address
- The protocols of keys exchange/generation of security associations must assume the complexity of *SCTP*. Thus, the work proposed in (Cano, 2011) recommends the construction of a new type *ID* for *ISAKMP*: *ID_LIST*, which represents a set of identities. However, using these lists of identities has its own drawbacks. For example, for *IKEv1*, a signature must be linked to a unique identity along all the same phase. But in the context of *SCTP*, the signer is not necessarily the same for each message. Accordingly, the signatory groups must share the same key, which involves security weaknesses in these practices on a large scale. Moreover, this work proposes an encoding multiple identities within a single certificate (for a single public key), but the support of this feature in the implementation of certification systems is dubious.

Another disadvantage of the use of *SCTP* with *IPsec* is that each *SCTP* packet is secured separately by *IPsec*. Hence, it increases the overhead when we have long messages that must be fragmented by *SCTP*,because several *SCTP* packets per message have to be secured.

Moreovere, there is a lack of efficiency in this security method that can decrease throughput and performance of the communication

## 2.2 SCTP-under-TLS

The use of TLS over SCTP is described in (Bellovin et al., 2003). *TLS* is currently mainly used on top of the *TCP*. But for *TLS* over *SCTP*, one *TLS* session must be established per stream. This leads to performance problems when many streams need to be secured. Every message is secured separately by *TLS*. Then, it is sent over *SCTP*. In case of sending many small messages, there will be an increased overhead compared to a solution that secures a complete *SCTP* packet containing several bundled messages.

## 2.3 Secure SCTP

To overcome the different problems of using *TLS* or *IPsec* to secure , Secure *SCTP* integrates cryptographic functions into SCTP (Jungmaier, A., Rescorlaand, E., Tuexen, M., 2002). Like *TLS* and *IPsec*, itprovides authentication, integrity and confidentiality since it uses the same standard cipher and *HMAC* algorithms as these standardized security solutions.

Nevertheless, *SSCTP* has a disadvantage compared to *TLS over SCTP*. Indeed, when long messages have to be fragmented at the *SCTP* layer, *TLS* secured firstly the whole message before fragmenting it. However, *SSCTP* has to secure each packet fragmented separately, which adds overhead. Moreover, *SSCTP* has to complete a secure session with messages and news chunks before securing data transmission, which causes more communication delay.

## 2.4 AUTH-SCTP

The extension presented in [4] provides a mechanism for deriving shared keys for each association. It defines a new chunk type, several parameters, and procedures for (*SCTP*). Authentication Chunk (*AUTH*) is the new chunk type added by this extention, which is used to authenticate *SCTP*. Random Parameter (*RANDOM*), Chunk List Parameter (*CHUNKS*)and Requested *HMAC* Algorithm Parameter (*HMAC-ALGO*)are the new parameters that are used to negotiate the authentication during association setup and establish the shared keys. However,authors in this work didnot definehow shared keys are exchanged. Another disadvantage of this extention is the increasing of the complexity of *SCTP* by adding new parameters, new chunk and proceduresthat add delay or degrad the quality of service.

# 3 SCTPAP SCHEME

In this paper, we propose the secure optimized authentication for *SCTP* (*SCTPAP*) scheme, which approaches the problem of the security during a node's authentication to connect for a first time to the network. The proposed algorithm uses an initialization phase to generate and exchange keys and public parameters recorded when the node wants access to the network for the first time. When the node obtains, at the end of this step, a secret key shared with the authentication server *AS*, it can connect with any legitimate node. If the connection is interrupted and the mobile node wants to re-connect with the same node, the procedure of re-authentication will be triggered.

In the proposed scheme, we assume that the network layer is secured by a tunnel *IPsec* and we protect the transport layer by the authentication procedure. The considered scenarios are between two nodes that should support symmetric and asymmetric encryption mechanisms. The proposed scheme uses the authentication server *AS* to achieve authentication procedure.

## 3.1 Initialization Phase: Node's Recording

The node must subscribe to the AS directly to gain access to the wireless network. We present the recording process as follows:

The node *A* computes its identity "*IDA*" by applying the hash functions on the concatenation of all its addresses. *IDA* is a unique identity of the node *A*. Node *A* generates a random number *x*. To send the two parameters x and *IDA* to *AS* server for recording, node *A* follows the following steps:

- It generates a random key *KS*
- It computes and sends *m1,* which is the encryption of key *KS* by the *AS* ' public key *serv-n*

  $m1= \{KS\}_{serv-n}$ *(1)*

- Then, it sends *m2* which is the encryption of x and *IDA* by this key *KS.*

After receiving *m1* and *m2*, AS decrypts m1 by its private key to get the key *KS* and decrypts *m2* by *KS* to get *x* and *IDA*. Then, *AS* selects randomly a number *y* and calculates *D,* which is the encryption *(AS-ID* || *y)* by the key *x*, as follows: *D = Ex (AS-ID* || *y) (2)*.

*AS* calculates the master key *K = (IDA* || *AS-ID* || *x* || *y) (3)* and sends the *D* to node *A*. This one decrypted *D* to get *AS-ID* and *y* and hence it can calculate the key *K*.

## 3.2 Initial Authentication

After the recording phase, the node *A*, wishing to connect with a node *B*, executes the initial authentication process, illustrated by Figure *1*.

As illustrated in this figure, after the establishment of a *IPsec* tunnel between the two nodes (step 1) and after the initialization phase of the *SCTP-AUTH* connection establishment in step 2 and step 3, node *A* follows these steps:

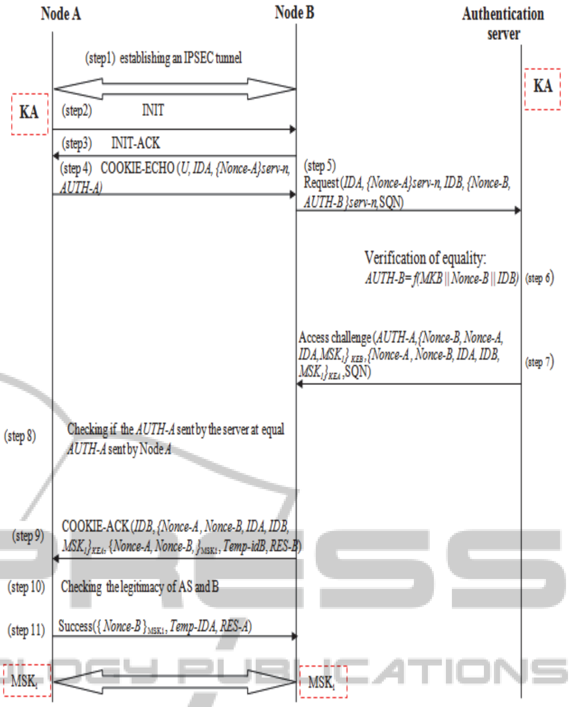

Node A
Node B
Authentication server
(step1) establishing an IPSEC tunnel
KA
KA
(step2) INIT
(step3) INIT-ACK
(step 4) COOKIE-ECHO (U, IDA, {Nonce-A}serv-n, AUTH-A)
(step 5) Request (IDA, {Nonce-A}serv-n, IDB, {Nonce-B, AUTH-B }serv-n, SQN)
Verification of equality: AUTH-B= f(MKB || Nonce-B || IDB)
(step 6)
Access challenge (AUTH-A, {Nonce-B, Nonce-A, IDA, MSK1} KEB, {Nonce-A , Nonce-B, IDA, IDB, MSK1}KEA , SQN)
(step 7)
(step 8)
Checking if the AUTH-A sent by the server at equal AUTH-A sent by Node A
(step 9)
COOKIE-ACK (IDB, {Nonce-A , Nonce-B, IDA, IDB, MSK1}KEA, {Nonce-A, Nonce-B, }MSK1 , Temp-idB, RES-B)
(step 10)
Checking the legitimacy of AS and B
(step 11)
Success({ Nonce-B }MSK1 , Temp-IDA, RES-A)
MSK1
MSK1

Figure 1: Initial authentication.

- It generates *Nonce-A*: a random number,
- It calculates *MKA = f (KA* || *Nonce-A)* where *f* is a hash function
- It calculates the *challenge Auth-A = f(MKA* || *Nonce-A* || *IDA)*
- It sends to node *B* a *COOKIE-ECHO,* which contains an *U* bit set to 0 informing that it is the initial authentication, its identity *IDA*, the random number *Nonce-A* encrypted by the server's public key*serv-n* and the challenge *AUTH-A* (step 4).

Node *B* sends a Request to *AS* (step 5) which contains *(IDB, {Nonce-B, AUTH-B }serv-n)* to prove its legitimacy to *AS* and *(IDA, {Nonce-A}serv-n* to compute *Auth-A* and send it to node *B*. After receiving these parameters, (step 6) *AS* calculates and verifies the equality *Auth-B = f (MKB* || *Nonce-B* ||*IDB)* to ensure the legitimacy of the node *B*. If the *Auth-B* calculated locally is equal to *Auth-B* received by the node *B*, *AS* generates the followings:

- *MKA= f(KA* || *Nonce-A)*
- *Auth-A = f(MKA* || *Nonce-A* || *IDA)*

- $MSK_1$ *= f (MKA* || *MKB*|| *NonceA* || *NonceB) which is a* session key for encrypting data that will be transmitted between node *A* and node *B* after the initial authentication phase.

In Step 7, the server sends an Access-Challenge to node B, which contains Auth-A, the set {Nonce-A, Nonce-B, IDA, IDB, MSK1} encrypted by the A's cipher key KEA and the set {Nonce-B, Nonce-A, IDA, MSK1} encrypted by the B's cipher key KEB. In Step 8, Node B compares Auth-A sent by the server with Auth-A sent by node A. If they are equal (Step 9), it sends a COOKIE-ACK containing the parameters *(IDB, {Nonce-A, Nonce-B, IDA, IDB, MSK1}KEA*, *{Nonce-A, Nonce-B}MSK1, Temp-idB, RES-B))* to node A. On receiving this message, (Step 10) node A finds the temporal ID of B which is Temp-idB with its digest RES-B=f(Temp-id-B||MSK1), then deciphers the encrypted parameters by its cipher key *KEA*. Finding the recent *Nonce-A* and *IDA*, it ensures the legitimacy of *AS* and finds the new session key *MSK1* with the *Nonce-B*. It will prove more its legitimacy to node *B* and that it has received *MSK1* by sending a success containing the Nonce-B encrypted by MSK1 and a temp-IDA with its RES-A for this new communication (Step 11). Decrypting the second set received from *B* $\{Nonce\text{-}A, Nonce\text{-}B, Temp\text{-}idB\}_{MSK1}$, node *A* verifies the legitimacy of *B* and that it has received *MSK1*. Both nodes calculate their temporary identities *(Temp-idA and Temp-idB)* that they will use during this session.

- *Temp-idA= f(IDA||SPI)*
- *Temp-idB= f(IDB||SPI)*

If node *A* changes its current address, it sends a status-chunk that contains the bit *U* set to *1* to inform *B* that its address has changed and consequently node *B* will change the destination address of the association between node *A* and node *B* in different databases *IPsec* to not establish a new *IPsec* association.

## 3.3 Re-Authentication between the Same Nodes

When a node *A* that is already connected to a node *B* is suddenly disconnected due a failure for example and then tries to connect again to the same node *B*, it must be re-authenticated. The re-authentication procedure is shown by Figure 2.

After establishing a channel *IPsec* tunnel between the two nodes and the initialization of the connection *SCTP-AUTH* (step 2), (step 3), and (step 4), node *A* sends the *U* bit set to 1 to inform that it is re-authenticating its previous temporary identity, the previous temporary identity of node *B*, a new *Nonce-A* and *Auth-A*. On receiving these parameters, node *B* notices that this is a re-authentication by examining the *U* bit. Then, the procedure of re-authentication begins by verifying the previous temporary identity of node*A* in its database. If it exists, it checks its temporary identity claimed by node *A*. If it is equal to its temporary identity that exists in its database associated with the previous association between it and node *A*, it calculates the *Auth-A = f (MSKi||Rand-A)* and (step 5) compares *Auth-A* locally computed with the one sent by node *A*. If they are equal, node B sends a *COOKIE-ACK,* which contains *(Auth-B, Nonce-B)* to authenticate node *A*. Node *A* calculates and verifies the *chall-B*. If the computed one is equal to that sent, then it sends a success message to allow a new association between these two nodes. Finally, the two nodes compute the new temporary identity (*Temp-idA+1* et *Temp-idB*+1) that they will use during this session, where:

- *Temp-idA+1= f(Temp-idA||SPI)*
- *Temp-idB= f(Temp-idB||SPI)*

and compute *MSKi +1 = f (MSKi||Rand-A||Rand-B).*The database in each node is updated at the end of re-authentication process, where the updated parameters are *SPI*, *Temp-idA*, *Temp-idB* and $MSK_i$.

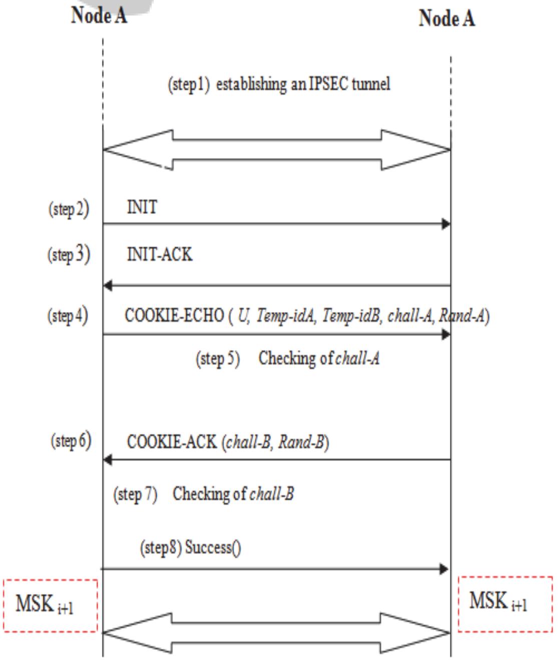

Node A
Node A
(step1) establishing an IPSEC tunnel
(step 2) INIT
(step 3) INIT-ACK
(step 4) COOKIE-ECHO ( U, Temp-idA, Temp-idB, chall-A, Rand-A)
(step 5) Checking of chall-A
(step 6) COOKIE-ACK (chall-B, Rand-B)
(step 7) Checking of chall-B
(step8) Success()
MSK i+1
MSK i+1

Figure 2: Re-authentication procedure between the same nodes.

# 4 ANALYSIS AND VALIDATION OF THE PROPOSED SCTPAP

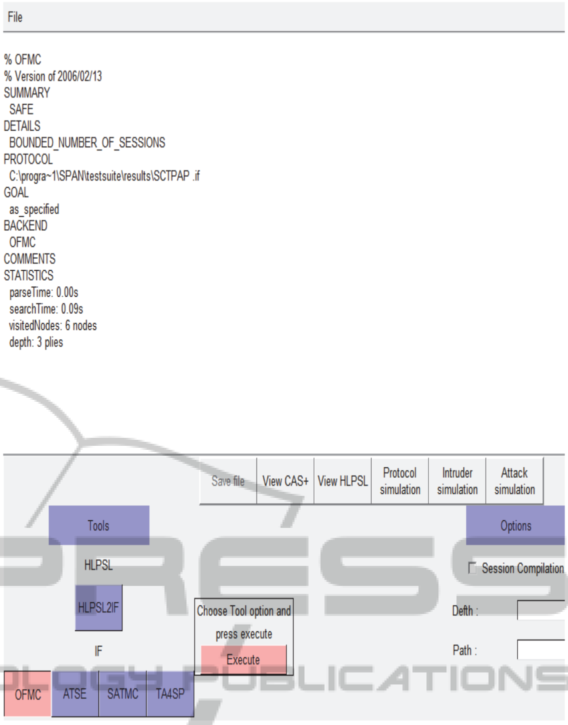
Our authentication protocol is a deliberate compromise between security and *QoS*. Indeed, the stronger the security is the higher delay of authentication is. However, increasing authentication's delay can interrupt the connection or degrades the *QoS*. Therefore, our scheme uses the most necessary parameters to protect against different attacks with lower delay of authentication.

This section presents security analyze of the proposed *SCTPAP* with the *AVISPA* (Automated Validation of Internet Security Protocols and Applications) tool (www.avispa project.org) without modeling tunnel *IPsec* in network layer. The tool attempts to detect attacks against protocols tested and tries to prove the validity of these protocols. High Level Protocol Specification Language (*HLPSL*) is a modeling language that *AVISPA* uses to write specifications for security protocols. We define three roles in our *HLPSL* specification of *SCTPAP*: *NodeA*, *NodeB* and *HAAA*. In each role, we specify its public and local parameters in addition to the messages sent and received by this role. We use the software *SPAN* (Security Protocol Animator) for *AVISPA* to verify the security of our protocol.

We can see the results of this verification by the *OFMC* (On-the-Fly Model-Checker) in figure 3 and *CL-ATSE* (Constraint-Logic-based Attack Searcher) in figure 4. Both of these *AVISPA* backends show that our protocol is safe as it is shown in figures 3 and 4.

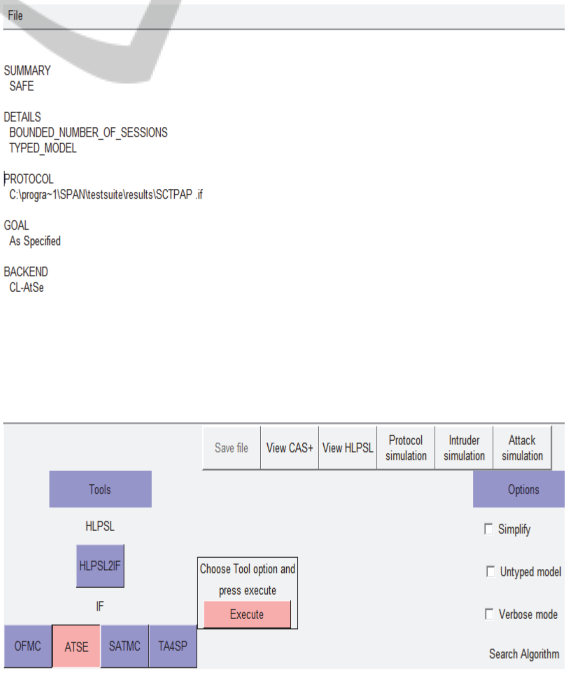
Span uses multiple attack scenarios to verify the security of the implemented protocol. Figure 5 shows the worst scenarios where the attacker captures wholes messages sent between the two nodes. However, we can see that all the messages captured by the attacker are neither modified nor exploited. Hence, our scheme is safe even against the worst attack scenarios. Indeed, all the parameters in each message captured are the same in the message, sent to the appropriate node. So the attacker can't:

- see the confidential parameters because they are encrypted,
- usurp (grab) the identity of any node or server, and
- modify the exchanged messages between the two nodes and the server *AS*.



```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  C:\progra~1\SPAN\testsuite\results\SCTPAP .if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.09s
  visitedNodes: 6 nodes
  depth: 3 plies
```

Figure 3: OFMC results.



```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  C:\progra~1\SPAN\testsuite\results\SCTPAP .if

GOAL
  As Specified

BACKEND
  CL-AtSe
```

Figure 4: ATSE results.

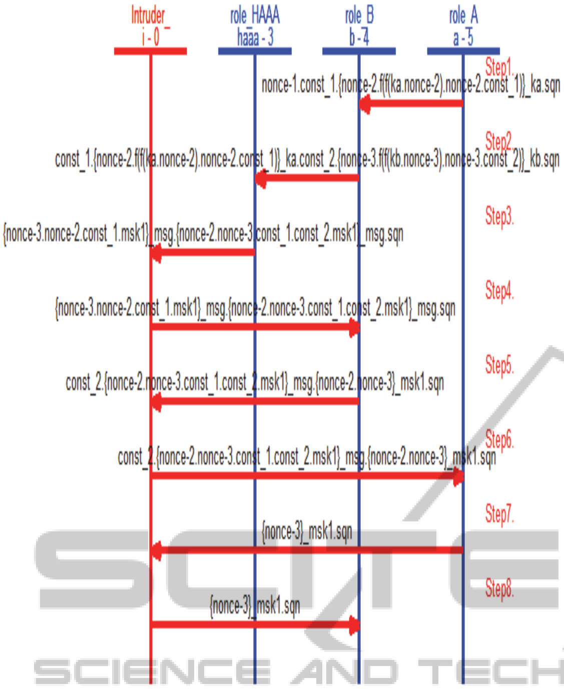

Intruder
i - 0
role HAAA
haaa - 3
role B
b - 4
role A
a - 5
Step1.
nonce-1.const_1.{nonce-2.f(f(ka.nonce-2).nonce-2.const_1)}_ka.sqn
Step2.
const_1.{nonce-2.f(f(ka.nonce-2).nonce-2.const_1)}_ka.const_2.{nonce-3.f(f(kb.nonce-3).nonce-3.const_2)}_kb.sqn
Step3.
{nonce-3.nonce-2.const_1.msk1}_msg.{nonce-2.nonce-3.const_1.const_2.msk1}_msg.sqn
Step4.
{nonce-3.nonce-2.const_1.msk1}_msg.{nonce-2.nonce-3.const_1.const_2.msk1}_msg.sqn
Step5.
const_2.{nonce-2.nonce-3.const_1.const_2.msk1}_msg.{nonce-2.nonce-3}_msk1.sqn
Step6.
const_2.{nonce-2.nonce-3.const_1.const_2.msk1}_msg.{nonce-2.nonce-3}_msk1.sqn
Step7.
{nonce-3}_msk1.sqn
Step8.
{nonce-3}_msk1.sqn

Figure 5: Attack scenario.

Not only *SCTPAP* is protected against different attacks according to the *SPAN* (safe), but also it has several strengths of security which are:

- Mutual Authentication: Nodes are mutually authenticated via the *AS* and each node is mutually authenticated with *AS*.
- Confidentiality: Not only the channel of communication is secured with tunnel *IPsec*, the confidential parameters are also encrypted by a dynamic cipher key and the Nonce of each node is sent ciphered to the *AS* by its public key.
- Integrity: The contents of the *Auth-A* and *Auth-B* with its *Nonce-A* and *Nonce-B* can't be modified by any malicious node.
- Degrees authentication: Mutual belief on the *MSK* key between *A* and *B*.

## 5 CONCLUSIONS

In this paper, we have performed an authentication protocol to secure a multi-homing connection between two nodes by keeping the same association *IPsec* when changing multi-homing *IP* addresses of these two nodes and by encrypting their communication with a dynamic session key. The proposed scheme, called *SCTPAP*, offers a compromise between security and *QoS*. In fact, with a minimum of messages and parameters, it protects the communicating nodes against attacks, where we used the SPAN tool to simulate the authentication procedure without the tunnel *IPsec* and we found it is safe.

The next step in the development of *SCTPAP* is to make it mobile suitable for heterogeneous wireless networks. Then, we will simulate the whole mechanism and qualitatively compare it with the other existing security solutions described in this paper. Moreover, the *QoS* evaluation will be more considered in future work.

## REFERENCES

Cano, M.D., 2011. "On the Use of SCTP in Wireless Networks, Recent Advances in Wireless Communications and Networks". Jia-Chin Lin (Ed .), ISBN: 978-953-307-274-6.

Bellovin, S., Ioannidis, J., Keromytis, A., Stewart, R., 2003. RFC3554: "On the Use of Stream Control Transmission Protocol (SCTP) with IPsec",July 2003. http://tools.ietf.org/html/rfc3554

Jungmaier, A., Rescorlaand, E., Tuexen, M., 2002. RFC 3436: "Transport Layer Security over Stream Control Transmission Protocol", December 2002. http://tools.ietf.org/html/rfc3436

Hohendorf, C., Unurkhaan, E., Dreibholz, T., 2006. "Secure SCTP draft-hohendorf-secure-sctp-02.txt", August 2006. http://tools.ietf.org/html/draft-hohendorf-secure-sctp-02

Tuexen, M., Stewart, R., Leiand, P., Rescorla, E., 2007. RFC 4895: "Authenticated Chunks for the Stream Control transmission Protocol (SCTP)", August 2007. http://www.ietf.org/rfc/rfc4895.txt

The avispa project. http://www.avispa project.org/

El Bouabidi, I., Zarai, F., Obaidat, M. S., Kamoun, L., 2014. "An efficient design and validation technique for secure handover between 3GPP LTE and WLANs systems" , *Journal of Systems and Software* (JSS), "Elsevier", Vol. 91, pp. 163-173, (Impact Factor = 1.135).

Samoui, S., El Bouabidi, I., Obaidat, M. S., Zarai, F., 2014. "Improved IPsec tunnel establishment for 3GPP–WLAN interworking", *International Journal of Communication Systems* (IJCS), "Wiley", Vol. 27, No. 2 (Impact Factor = 0.712)

El Bouabidi, I., Zarai, F., Obaidat, M. S., Kamoun, L., 2012. "Secure Host-based Mobility Protocol for Wireless Heterogeneous Networks" Proceedings of the *12th IEEE International Conference on Scalable Computing and Communications* (ScalCom 2012), Changzhoun, Chine, 17-19.

Smaoui, S., Zarai, F., Obaidat, M. S., Kamoun, L., 2012. "Authentication Optimization for Vertical Handover in Heterogeneous Wireless Networks," Proceedings of the 2012 International Conference on Wireless Information Networks and Systems, WINSYS 2012- Part of ICETE 2012, 2012, pp. 249-254.