Exploring a Security Protocol for Secure Service Migration in Commercial Cloud Environments

ABSTRACT

Mobile users are making more demands of networks. They want to run applications such as network steaming of audio and video as well as immersive gaming that demand high Qualities-of-Service. One way to address this problem is by using mobile services that move around as users move around. This ensures that low latencies are maintained between the client and the server resulting in a better Quality-of-Experience. In addition, the advent of virtual machine technology for example, VMware, and container technology, such as Docker, have made the migration of services between different Cloud Systems possible. Furthermore, a Service Oriented Architecture that supports service migration in Cloud environments has been proposed. Though there are many things in place to support mobile services, a key component that is missing is the development of security protocols that allow the safe transfer of servers to different Cloud environments. In particular, it is important that servers do not end up being hosted on unsafe Cloud infrastructure and also Clouds do not end up hosting malicious servers. This paper proposes a new security protocol to address this issue. The protocol is specified and tested using AVISPA. The initial results indicate that the protocol is safe and therefore can be used in practical systems.

Keywords

Future Internet, Y-Comm Framework, Mobile Services, Cloud Computing, Security Protocols, AVISPA.

1. INTRODUCTION

Cloud computing is an emerging technology and along with mobile services, is rapidly becoming the next Internet-based enterprise platform. Currently, data owners are migrating their data to cloud storage platforms without buying any storage devices. This gives many benefits to the large enterprises as well as to individual users because it reduces the cost of storage services and allows these services to be managed in a more dynamic way. Cloud computing is, therefore, facilitating the migration of data and services. These services are called mobile services and will be used to support mobile users as they move around. These services can migrate between clouds to reduce the latency between the service and the mobile users, hence maintaining a good Quality of Service (QoS) to users. For example, two people are watching a video in London; if they then take a train from London to Cambridge, the video service will notice that the increasing latency between the users and the server and may choose to replicate the service on Cloud systems nearer to Cambridge. In addition, in this brave new world, Cloud Providers will actively advertise their Cloud resources to mobile services which will use these advertisements to dynamically migrate its servers to these Clouds.

In this new environment, mobile users will demand to be always connected using heterogeneous networking. Mobile devices will, therefore, have several wireless interfaces including Wi-Fi, LTE, 5G, satellite and Ultra-wideband interfaces. These networks will seamlessly work together using vertical handover techniques [14]. The Y-Comm architecture [15] has been developed to build future mobile systems that can provide seamless communications. Hence, future mobile systems will need to support both mobile users and mobile services. Though extensive work has been done to support mobile users, less work has been done on the development of mobile services. However, there is now increasing interest in this area as there is a need to provide services at the edge of the network, i.e. to support edge-computing. One aspect of the research on mobile services that has been inadequate is support for security [5]. In particular, it is important that servers do not end up being hosted on unsafe Cloud infrastructure which can hamper service delivery to mobile clients and also Clouds do not end up hosting malicious servers which can damage Cloud infrastructure. This paper attempts to address these issue by providing a new security protocol for secure service migration. This security protocol is specified for service migration over the Cloud and is tested through a simulation study using the Automated Validation of Internet Security Protocols and Applications (AVISPA) tool which is used for the analysis of large-scale Internet security sensitive protocols and applications.

The rest of the paper is organized as follows. Section II presents the related work; Section III describes background methodologies for our solution approach. Section IV details our solution approach while Section V shows the results for the test scenario. The paper concludes with Section VI.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

2. RELATED WORK

2.1 Migration of services

Service migration has been proposed for many environments and is increasingly being used in Cloud environments that support virtualisation. This is possible because the virtual machine paradigm allows entire virtual machines to be migrated. Virtual machine migration can be expensive as the entire virtual machine has to be moved. The emergence of container technology, such as Docker [1], in which containers housing several services can be migrated is gaining in prominence[reference]. Unikernels [11] in which the operating system is bounded and customized to run a single main application is the next emerging specimen in this genre and should, from a management point-of-view, make server migration simpler. However, these efforts assume that the communication architecture does not help facilitate server migration which makes these mechanisms difficult to use in Wide Area Networks (WANs).

2.2 The Y-Comm Framework

Y-Comm is an architecture that has been designed to build heterogeneous mobile networks. It attempts to integrate communications, mobility, QoS and security into a single platform. It divides the Future Internet into two frameworks: the Core Framework and the Peripheral Framework. The researchers of Y-Comm have made major contributions in the areas of proactive handover [13] as well as introducing new concepts in security such as Targeted Security Models [12]. Other network architectures for mobile systems such as Hokey [9], Ambient Networks [18] and Mobile Ethernet [16] have also been explored. HoKey looked at issues of secure handover in heterogeneous networks while Ambient Networks concentrated on supporting seamless connectivity in diverse networks. Mobile Ethernet adopted the Core/Peripheral structure like Y-Comm but assumed an Ethernet-type Core. A comparison of these systems indicate that Y-Comm offers the most functionality and flexibility [3] while integrating various key mechanisms [2], [4].

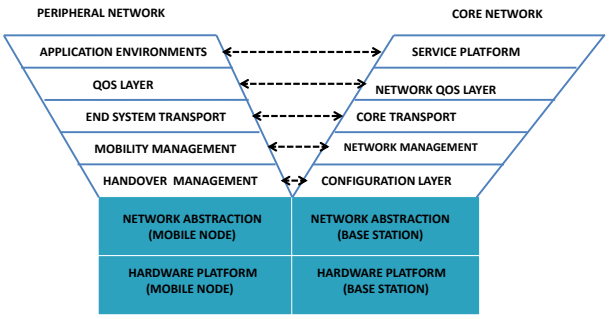

Figure 1: The Y-Comm Framework: The Reference Model

The Y-Comm Reference Model is shown in Figure 1. A more detailed description of the Y-Comm Framework is given in [15]. However, it should be noted that Y-Comm supports a layer for mobile services called the Service Platform Layer in the Core Network. This layer allows services to be installed and managed coherently in future networks including support for the mobile services.

2.3 A Service-Oriented Framework for Mobile Services

In order to provide a complete set of mechanisms to enable mobile services, it is necessary to develop a new Service oriented Architecture that allows services to be managed, copied or migrated to support mobile users [6]. The system should also provide algorithms that incorporate traffic management as well as the QoS requirements of the flow. This new Framework was proposed in [10] and has six layers as shown in Figure 2, which, from the top to the bottom, are briefly described below:

- **The Service Management Layer**: This layer manages the service that is being provided. It specifies the functions of the service, registers the service in a service registry and obtains a unique Service_ID. It also specifies the minimum resources required by networking and cloud infrastructure needed to run the service in terms of computing resources, network Quality-of-Service requirements and storage needs.

- **The Service Subscription Layer**: This layer handles the functions required for global clients to use the service. It therefore allows clients to subscribe to services. It provides the user with a unique Client_ID, a given Service Level Agreement (SLA) and sets up accounting and payment mechanisms.

- **The Service Delivery Layer**: The layer is in charge of delivering the service to a given client. It first maps the SLA to a given QoS and ensures that the selected server as well as its networks can meet the required QoS. The service also receives notifications and triggers about handovers and based on these notifications, may replicate or migrate the service closer to the user.

- **The Service Migration Layer**: The layer handles the replication or migration of services to different cloud platforms to facilitate a good QoE for the mobile user. Migration is done at the request of the Service Delivery Layer.

- **The Service Connection Layer**: This layer is responsible for managing the connection between a client and the service and reports changes in transport or network parameters such as bandwidth or delay to the Service Delivery Layer.

- **The Network Abstraction Layer**: This layer allows the service to interface to different types of networks, depending on network architecture and addressing. In the current Internet, this layer maps onto IP networking with TCP/IP. In more advanced systems such as Y-Comm, this functionality is spread between the QoS and Transport Layers in Core and Peripheral Frameworks.

This is a powerful framework that can be used to allow services to migrate from one Cloud to another Cloud. The work of Sardis [19] showed that in order to migrate a service it is necessary to compare the time taken to migrate the service with the amount of time the user will be in the region concerned. Hence, the mobility model of the user must be considered. Sardis used a simple queuing model to represent user mobility in mobile networks.

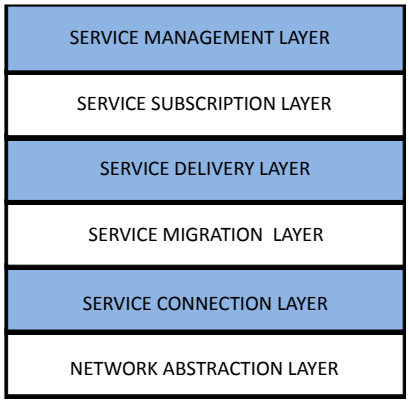

Figure 2: Service-Oriented Framework for Mobile Services

2.4 The emergence of Edge Computing

The desire to localise services has led to the emergence of edge computing where compute servers are placed on Cloud close to the edge rather than the centre of the Internet [20], [8]. This reduces latency and provides high bandwidth via the local network. Edge computing also enables support for networks that provide highly mobile environments such as Vehicular Ad-Hoc Networks (VANETs). In addition, mobile edge computing is also being used to support communication using 5G Cloud Enabled Small Cells [7].

Though all these mechanisms are promising developments, it is necessary to consider security as part of the overall design. This paper addresses this issue in the context of the Service-Oriented Framework described above.

3. SPECIFY THE SECURITY PROTOCOL FOR MOBILE SERVICES

In this section, we look at protocols for secure service migration. As stated previously, the main issues are fraudulent Cloud providers that entice the service providers to host their services on faulty Cloud facilities resolving in data loss and lack of service as well as misbehaving services which convince cloud providers that they are well-behaved systems leading to mismanagement and abuse of Cloud facilities. We focus on developing a security protocol that handles these issues.

A key component of these mechanisms is the use of a Registry. A Registry or Certificate Authority (CA) is a trusted party that issues signed electronic documents to verify whether the party is a valid entity on the Internet. Electronic documents used as digital certificates are important in public key encryption(PKE), usually, these certificates include the owner's name, public key, the expiration date of the digital certificate, location of the owner and other data of the owner. CA lists will be maintained on operating systems and browsers. The Registry contains secure information about services including Cloud facilities. In addition, the registry can talk securely to both services and Clouds using public key encryption. All service and cloud providers must register their services with the Registry. The Registry, therefore, has public keys of all Cloud Systems and Servers in the systems. In this section, we specify two situations: the first is when the server migrates from home network to a cloud. The second is service migrate between two clouds. Below we look at these situations in turn.

3.1 General Notation

3.1.1 The Server

The server (S) as represented by:
Server_ID, Type of service (TOS), Public Key (PKS).
The server is identified by a unique Service_Id which is given to the server when it registers with the Registry. The server is located at a place denoted by X, when the server is located in its home network X= H; X = A when the server has migrated to Cloud A and B for Cloud B.

3.2 Cloud Facilties

The Cloud facilities are represented as follows:
Cloud A (CA) = Cloud_ID_A, TOS = Cloud, PKA, Resources.
Cloud B (CB) = Cloud_ID_B, TOS = Cloud, PKB, Resources.
Each Cloud is uniquely identified by a Cloud_ID and since they are cloud services, TOS = Cloud; PKA and PKB are public keys for Cloud A and Cloud B respectively. In addition, each Cloud will have a number of resources which it actively advertises to servers.

3.2.1 The Registry

The Registry is the last key component and is used to verify the identities of all servers on the network. In addition, the Registry knows the Service_IDs and Public Keys for each service.

We represent the public key for the registry as PKR.
Registry(R) = PKR

3.3 Nonces denoted by N

Nonces are randomly generated numbers which are unforgeable and are used as session tokens, ensuring that requests cannot be repaid by unauthorised personnel.

3.4 Timestamps denoted by T

Timestamps are used explicitly with migration requests and responses. This allows the system to evaluate how long it takes for requests to be granted and how long it takes for migration transfers to complete. In the protocol, timestamps are represented as simple strings.

4. MIGRATION FROM CLOUD A TO CLOUD B

In this section, we look at when the service moved from Cloud A to Cloud B. The protocol is followed in exactly the same way as outlined above:

The full explanation of this second migration is given below:

- **Step 1** The service on Cloud A receives advertisements from Cloud B advertising Cloud's B resources.
- **Step 2** The Server checks the validity of Cloud B.
- **Step 3** Registry authenticates Cloud B
- **Step 4** The server on Cloud A sends a request to migrate to Cloud B

Algorithm 1 Security Protocol for Migration between Cloud A to Cloud B

1: CB → SA: Advertisement ($CloudID\_B$, TOS, Resources, PKB)
2: SA → R: Verify Identity ($CloudID\_B$, TOS, PKB, Resource, Server_ID, PKS) PKR
3: R → SA: Message: YES ($CloudID\_B$, TOS, PKB, Valid Resources) PKS
4: SA → CB: Migration Request + (Server_ID, TOS, TA, Req.Resource, PKS, NA) PKB
5: CB → R: Verify Identity (Server_ID, PKS, TOS, Cloud ID_B, PKB) PKR
6: R → CB: Message: Yes (Server_ID, TOS, Valid Service) PKB
7: CB → SA: Migration Response + (Cloud ID_B, TOS= Cloud), TB, Resources Granted, NA, NB) PKS
8: SA → CB: Transfer (Migration) + (Server_ID, $CloudID\_B$, Services, NB) PKB
9: CB → SA: $Transfer\_Ack$ (Server_ID, $CloudID\_B$, NA, Tcomp) PKS
10: SB → R: Transfer-Complete + (Server_ID, $CloudID\_B$, TOS, TA, TB, Tcomp) PKR

- **Step 5** Cloud B sends a request to make sure that the server, S, is a valid service:
- **Step 6** The Registry validates Cloud B
- **Step 7** Cloud B signals to The Service on Cloud A that it is OK to migrate the service to Cloud B:
- **Step 8** The Service signals to Cloud B to migrate the service:
- **Step 9** Cloud B signals to the server on Cloud A that the migration is complete: Hence, **New Location: (SA→SB)** The service is now started on Cloud B.
- **Step 10** The service on Cloud B signals to the Registry that the migration has been completed:

Figure 3 shows the steps for Cloud to Cloud migration of services.

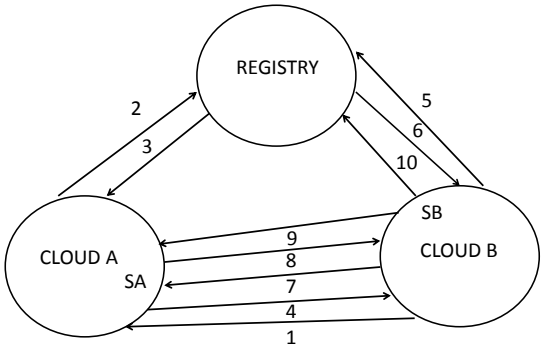

Figure 3: Migration from Cloud A to Cloud B

4.1 Key Observations

There are important observations to be made with this protocol. Firstly, nonces are used to protect this session between the Server and the Cloud to which the server wants to migrate.This is done using an extended approach based on the Needham Schroeder protocol [17]. This is seen in the following steps:

- In Step 4, the Server uses NA as part of the transfer request .
- In Step 7, Cloud B replies to the Server using NA and NB as part of the transfer response.
- In Step 8, Server authorises the transfer of the service to Cloud B using NB.
- In Step 9, Cloud B signals that the transfer is completed using NA.

It is also essential to understand Step 10 in which the new transferred service reports back to Registry about its new location. This allows the Registry to be able to track server migration in response to user needs and therefore can be used to do more effective planning of server provision which will cope with user mobility. From a security point of view this is better than having to track individual users. In addition, by submitting the timestamps, it is possible to find out how much time it took for Cloud A to reply to the Server's migration request (TB - TA), as well as the time taken to do the actual transfer (Tcomp - TB). These values are used by the Registry to track the migration performance of individual Clouds. The Registry therefore is being used to give insight into the performance of the infrastructure.

4.2 Using AVISPAs

In this section, AVISPA is used to analyse the protocol specified in the previous section. AVISPA provides automated validation of Internet security protocols and applications. We propose the use of formal validation methods and model checking for security properties of these protocol. Here, we represent that formal specification and verification are carried out using the role based language , HLPSL(See Appendix) and AVISPA model checker which automates the model checking of the protocol and uses the parameters necessary for the formal verification of secure service for these protocols. In simple, we modelled in HLPSL and analysed with Avispa tool. It supports for multiple backend tools such as On_the_ ïňĆy Model-Checker (OFMC), Constraint-Logic-based Attack Searcher (CL-AtSe), SAT-based Model-Checker (SATMC), and Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP) and it verifies the security properties for a bounded number of sessions. When we develop a secure service migration protocol, it is important to prove that it has the expected results by the security properties or parameters. In HLPSL, to model the system refers that model the components of the security properties. Each backend tool has its own options and components to define the formal verification. After the execution process, the output describes the results of the protocol. The output is listing the common format of Summary, Details of bounded number of sessions and the goals. The AVISPA model checker, a widely accepted tool, provides the results of whether the security protocol is SAFE or UNSAFE.

4.3 HLPSL

We know that the AVISPA framework is role based and it gives more important to roles rather than exchanging information. In order to verify the security properties of the security protocol, we had to model three roles: Cloud A, Cloud B and the Registry, describing the actions of protocol. The roles can declare a set of local variables before the initialization of the model checking section, this initialization section is followed by the state transition section. HLPSL defines two composed roles: the session role and the environment role. The HLPSL provides the security properties in GOAL section.

4.4 OFMC and ATSE

After completing the specification as described above, we divided the verification process into two steps. The first step was to validate the specification using OFMC back-end tool of AVISPA, and the second step was to use ATSE back-end. In our protocol, AVISPA Outputs SAFE from OFMC and ATSE. The first back end tool is called OFMC, On-the-fly-model checker, verifies that combination of two main concepts. In the summary section of the results shown in Figures 4 and 5, based on the protocol specification, indicates that the protocol is SAFE. The second back end tool is called ATSE, Constraint-Logic-based Attack Searcher (CL-AtSe), also indicates that the protocol is SAFE. Now, both of the protocol results SAFE, so that the probability of the expected result is accomplished.

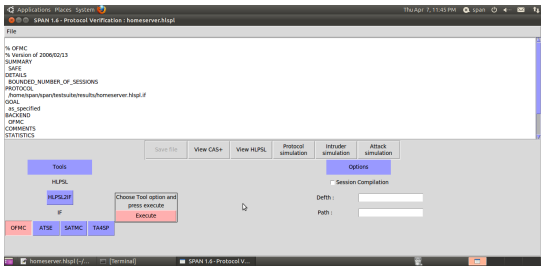

Figure 4: OFMC: Cloud A to Cloud B

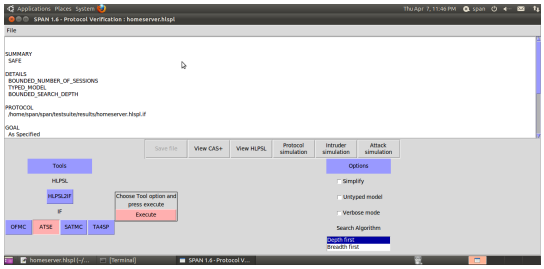

Figure 5: ATSE: Cloud A to Cloud B

5. FUTURE WORK

Using AVISPA tool, we are able to show that the protocol specified is safe in normal operation. More work is being done to analyse intruder attacks to show that the proposed protocol is secure against the active and passive attacks including replay and man-in-the-middle attacks.

6. CONCLUSIONS

This paper has presented a new security protocol for server migration between commercial Cloud environments. The system uses a Registry which validates servers and Cloud infrastructure as well as monitors the migration pattern of services to give an indication of user demand. In addition, the system records the responses of the Cloud Infrastructure to server requests. The protocol was verified using AVISPA. More work is being done to make this protocol resistant to intruder attacks.The protocol is safe under normal operation; the present protocol critically prevents impersonation attacks either by rogue cloud infrastructure hoping to sneer valid services or by malicious servers wanting inflict damage on Cloud infrastructure. We are exploring how this protocol and be enhanced to prevent intruder and man-in-the-middle attacks.

7. REFERENCES

[1] Docket technology, 2016-11-29.

[2] M. Aiash, G. Mapp, A. Lasebae, J. Loo, and R. Phan. A Formally Verified AKA Protocol For Vertical Handover in Heterogeneous Environments using Casper/FDR. *EURASIP Journal on Wireless Communincations and Networking (Open Springer)*, April 2012.

[3] M. Aiash, G. Mapp, A. Lasebae, J. Loo, F. Sardis, R. Phan, M. Augusto, R. Vanni, and E. Moreira. A Survey of Potential Architectures for Communication in Heterogeneous Networks. In *Proceedings of Wireless Telecommunications Symposium (WTS 2012)*, April 2012.

[4] M. Aiash, G. Mapp, A. Lasebae, R. Phan, and J. Loo. Integrating Mobility, Quality-of-Service and Security in Future Mobile Networks. *Electrical Engineering and Intelligent Systems: Lecture Notes in Electical Engineering*, 130:195–206, 2013.

[5] M. G. Aiash M and G. O. Secure Live Migration: Issues and Solutions. In *Proceedings of the Conference on Advanced Information Networking and Applications, AINA 2014, Victoria, Canada*, May 2014.

[6] Q. Duan, Y. Yan, and A. Vasilakos. A Survey on Service-Oriented Network Virtualization toward Convergence on Networking and Cloud Computing. *IEEE Transactions on Network and Service Management*, 9:373–392, 2012.

[7] F. J. et al. Introduction Mobile Edge Computing Capabilities through distributed 5G Cloud Enabled Clouyd Cells. *Mobile Networks and Applications*, 21(4), August 2016.

[8] S. F, M. G.E, L. J, and A. M. Dynamic Trsffic Management for Interactive Cloud Services: Localising Traffic Based on Network Throughput and User Mobility. In *Proceedings of the 7th IEEE/ACM International Conference on Utility and Cloud Computing*, December 2014.

[9] HoKey Working Group. Hokey: Security Support for IEEE 802.21: Media Independent Handover Services, 2010.

[10] S. M, M. G. E, L. J, A. M, and V. A. On the Investigation of Cloud-based Mobile Media Environments with Service-Populating and QoS-aware

Mechanisms. *IEEE Transactions on Multimedia*, January 2013.

[11] A. Madhavapeddy, R. Mortier, C. Rotsos, D. Scott, B. Singh, T. Gazagnaire, S. Smith, S. Hand, and J. Crowcroft. Unikernels: Library operating systems for the cloud. *SIGPLAN Not.*, 48(4):461–472, Mar. 2013.

[12] G. Mapp, M. Aiash, A. Lasebae, and R. Phan. Security Models for Heterogeneous Networking. In *International Conference on Secuity and Cryptography (SECRYPT 2010) Athens, Greece*, July 2010.

[13] G. Mapp, F. Katsriku, M. Aiash, N. Chinnam, R. Lopes, E. Moreira, R. P. Vanni, and M. Augusto. Exploiting Location and Contextual Information to Develop a Comprehensive Framework for Proactive Handover in Hetegeneous Environments. *Journal of Computer Networks and Communications*, February 2012.

[14] G. Mapp, F. Shaikh, M. Aiash, R. P. Vanni, M. Augusto, and E. Moreira. Exploring efficient imperative handover mechanisms for heterogeneous wireless networks. In *2009 International Conference on Network-Based Information Systems*, pages 286–291. IEEE, 2009.

[15] G. Mapp, F. Shaikh, J. Crowcroft, D. Cottingham, and J. Baliosian. Y-Comm: A Global Architecture for Heterogeneous Networking (Invited Paper). In *3rd Annual International Wireless Internet Conference (WICON)*, October 2007.

[16] K. Masahiro, Y. Mariko, O. Ryoji, K. Shinsaku, and T. Tanaka. Secure service and network framework for mobile ethernet. *Wireless Personal Communications*, 29, 2004.

[17] C. A. Meadows. *Analyzing the Needham-Schroeder public key protocol: A comparison of two approaches*, pages 351–364. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.

[18] N. Niebert, A. Schieder, A. Abramowicz, H. Malmgen, G. Sachs, J. Horn, C. Prehofer, and H. Karl. Ambient Networks: An Architecture for Communication Networks Beyond 3G. In *IEEE Wireless Communications*, volume 11, 2004.

[19] F. Sardis. *Exploring Traffic and QoS Management mechanisms to support mobile cloud computing using service localization in heterogeneous environments.* School of Science and Technology, Middlesex University, August 2014. PhD Thesis.

[20] F. Sardis, G. Mapp, J. Loo, and M. Aiash. Dynamic edge-caching for mobile users: Minimising inter-as traffic by moving cloud services and vms. In *Proceedings of the Conference on Advanced Information Networking and Applications, AINA 2014, Victoria, Canada*, May 2014.

APPENDIX

This is HLPSL file for the new security protocol.

```
role clouda_A( A,B,R : agent, PKA : public_key, SND_R,SND_CB,RCV_R,RCV_CB:
channel (dy)) played_by A def= local State:nat,
    NA,NB,TA,TB,TOS,Tcomp,Verify : text,
    Services,Succ,Ack,Resources,Advertisement : message,
    PKS,PKB,PKR : public_key,
    Sec_Services,CloudID_A,CloudID_B,Server_ID,
    Migrequest,Migresponse : protocol_id
    init

State := 1
    transition
    2.   State = 1  /\RCV_CB(Advertisement.CloudID_B.TOS'.Resources'.PKB')
=|> State':= 3 /\Server_ID':=new() /\PKS':=new() /\SND_R(Verify.CloudID_B.TOS'.PKB'.Resour
Server_ID'.PKS'_PKR)

4. State = 3 /\RCV_R(Succ.CloudID_B.TOS'.PKB'.Resources'_PKS)=|> State':=
5 /\TA':=new(),NA':=new() /\SND_CB(Migrequest.Server_ID'.TOS'.TA'.Resources'.PKS'.NA'_PKI

8. State =5 /\RCV_CB(Migresponse.CloudID_B.TOS'.TB'.Resources'.NA'.NB'_PKS)=|>
State':=7 /\SND_CB(Server_ID'.CloudID_B.Services.NB'_PKB) /\secret(PKS,Sec_Services,A,B)

10. State = 7 /\RCV_CB(Ack.Server_ID'.CloudID_B.NA'.Tcomp'_PKS) =|> State'
:= 9
    end role

role cloudb_B( A,B,R : agent,
    PKB : public_key,
    SND_SA,SND_R,RCV_SA,RCV_R : channel (dy))
    played_by B
    def=
    local
    State:nat,
    NA,NB,TA,TB,TOS,Tcomp,Verify : text,
    Services,Succ,Ack,Resources,Advertisement : message,
    PKS,PKA,PKR : public_key,
    Migrequest, Migresponse,Sec_Services,CloudID_A,CloudID_B,Server_ID : pro-
tocol_id,
    init
    State := 0
    transition
1. State = 0 /\RCV_SA(start) =|> State':= 2 /\TOS':=new(),Resources':=new(),PKB':=new()
/\SND_SA(Advertisement.CloudID_B.TOS'.Resources'.PKB')

5. State = 2 /\RCV_SA(Migrequest.Server_ID'.TOS'.TA'.Resources'.PKS'.NA'_PKB)
=|> State':=4 /\SND_R(Verify.Server_ID'.PKS'.TOS'.CloudID_B.PKB'_PKR)

7. State = 4 /\RCV_R(Succ.Server_ID'.TOS'.Services_PKB)=|> State':=6 /\NB':=new(),TB':=new
/\SND_SA(Migresponse.CloudID_B.TOS'.TB'.Resources'.NA'.NB_PKS)

9.  State =6 /\RCV_SA(Server_ID'.CloudID_B.Services.NB'_PKB)=|> State':=8
/\Tcomp' :=new() /\SND_SA(Ack.Server_ID'.CloudID_B.NA'.Tcomp'_PKS) /\SND_R(Server_ID'.Cl

end role
    role registry_R( A,B,R : agent,
    PKS,PKA,PKB : public_key,

SND_CA,RCV_CA,RCV_CB,SND_CB,SND_SA,
    RCV_SA,RCV_SB : channel(dy))
    played_by R
    def=
    local
    State:nat,
    Tcomp,TOS,NA,NB,NH,TA,TB : text,
    Services,Succ,Resources : message,
    PKR : public_key,
    Services,Verify: text,
    Sec_Services,CloudID_A,CloudID_B,Server_ID : protocol_id
    init
    State := 10
    transition

3. State = 10 /\RCV_SA(Verify.CloudID_B.TOS'.PKB'.Resources'.Server_ID'.PKS'_PKR)
=|> State':= 12 /\SND_SA(Succ.CloudID_B.TOS'.PKB'.Resources'_PKS)

6. State =12 /\RCV_CB(Verify.Server_ID'.PKS'.TOS'.CloudID_B.PKB'_PKR) =|>
State':= 14 /\SND_CB(Succ'.Server_ID'.TOS'.Services_PKB)

11. State = 14 /\RCV_SB(Tcomp.Server_ID'.CloudID_B.TOS'.TA'.TB'Tcomp'_PKR)=|>
State':=16

end role

role session( A,B,R : agent,
    PKS,PKA,PKB,PKR : public_key)
    def=
    local SND_CA, RCV_CA,SND_CB, RCV_CB,SND_R, RCV_R,
    SND_SA,RCV_SA,RCV_SB:channel(dy)
composition

clouda_A(A,B,R,PKA,SND_R,RCV_R,SND_CB,RCV_CB) /\cloudb_B(B,A,R,PKB,SND_CA,SND_R,R
/\registry_R(A,B,R,PKS,PKA,PKB,SND_CA,RCV_CA, RCV_CB,SND_CB,SND_SA,RCV_SA,RCV_SE


    end role

role environment()
    def=
    const
    a,b,r : agent,
    pka,pks,pkb,pkr : public_key
    composition
    session(a,b,r,pks,pka,pkr,pkb) /\session(b,a,r,pks,pka,pkr,pkb) end role

goal secrecy_of sec_services end goal


    environment()
```