# Authentication and Authorization Design in Honeybee Computing

Nur Husna Azizul[1], Abdullah Mohd Zin[2], Ravie Chandren Muniyandi[3], Zarina Shukur[4]
Center for Software Technology and Management (Softam)
Universiti Kebangsaan Malaysia, 43600 UKM Bangi, Selangor, Malaysia

***Abstract*—Honeybee computing is a concept based on advanced ubiquitous computing technology to support Smart City Smart Village (SCSV) initiatives. Advanced ubiquitous computing is a computing environment that contains many devices. There are two types of communication within Honeybee computing: client server and peer-to-peer. One of the authorization techniques is the OAuth technique, where a user can access an application without creating an account and can be accessed from multiple devices. OAuth is suitable to control the limited access of resources to the server. The server use REST API as web service to publish data from resources. However since Honeybee computing also supports peer-to-peer communication, security problem can still be an issue. In this paper, we want to propose the design of a secure data transmission for Honeybee computing by adopting the authorization process of OAuth 2.0 and Elliptic Curve Diffie-Hellman (ECDH) with HMAC-Sha. This article will also discuss the communication flow after adopting OAuth 2.0 and ECDH to the computing environment.**

***Keywords*—HMAC-Sha; REST API; peer-to-peer; web service; honeybee computing***

## I. Introduction

Honeybee computing is a concept based on advanced ubiquitous computing technology to support Smart City Smart Village (SCSV) initiatives1. It is supported by a middleware together with a number of tools such as semantic knowledge tool and predictive analytics for information management. The sources of information in Honeybee Computing are from the web, public and private cloud, and user devices. Since there is a multiple sources of data, it is important that all transactions are secured.

In the development of a software, the effort to secure the software is important, for example a framework cyber security [1] strategy framework is to protect government data, foreign investment and citizens. With many types of attacks, the importance of security is not only to look at securing the data but also to ensure users authenticity [2], especially if the interaction involves third party users. For example, a design for a virtual private network [3] for collaboration specialist users where the authentication becomes the main part of the design and authentication mechanism for an ad-hoc network [4]. One of the popular security problems within a network is the man-in-the-middle (MITM) attack [5][6]. The problem of a MITM attack is more critical in applications that use the single sign on (SSO) method. The Facebook platform that is based on cloud computing is open to multiple types of MITM attacks [5][7].

Authorization and authentication [8][9] are security issues that must be considered during the development of an application. There are multiple cloud service providers with client authentication method, for example Amazon Web services that use HMAC-Sha1, HMAC-Sha256, or X.509 certificate, Azure uses SAML 2.0 or Auth 2.0, Azure Storage uses HMAC-Sha256, and Google App Engine uses OAuth 2.0, shared secret or certificate. HTTP authentication [6][10][11] provides basic and digest access authentication.

Honeybee computing needed authorization authentication that support the architecture, since the Honeybee computing support peer-to-peer and client server, secure communication during data transfer is important to protect the resource. Client server use authorization and authentication that involved storing of key in server side, while peer to peer security mechanism usually involved with encryption and decryption. There is no security method for secure communication with both client server and peer to peer communication. This paper discusses the authorization and authentication process in Honeybee computing.

The rest of this paper is organized as follows: The existing work of the attack, client server and peer to peer method to secure the communication is presented in Section 2. In Section 3, we present the overview of Honeybee Computing, before discussing the findings in Section 4. Section 5 presents the communication flow. Finally, Section 6 concludes the paper and presents the future work.

## II. Related Work

### A. MITM Attack

Generally, there are three types of MITM attack, namely Address Resolution Protocol (ARP) Cache Poisoning, Domain Name System (DNS) Spoofing and Session Hijacking [12]. The attacks that use ARP spoofing [13] refers to a technique that enables an attacker to pretend to be one of the users in a communication between two users. The DNS spoofing principle [13] is where the victim's HTTP traffic is intercepted. The program analyzes incoming HTTPS links and replaces them with unprotected HTTP links or homographic ally related secure links. Session hijacking is the hijacking of a valid computer session to the browser. The aim of MITM [12] is to compromise the confidentiality, integrity and availability of messages. Based on the three effects, the scenario that would be caused by MITM would be as follows:

[1]GSIAC Smart City. " Smart City-Smart Village". http://gsiac.org/index.cfm?&menuid=36#sthash.74DYwYR0.dpuf [28 January 2015].

Confidentiality: The confidentiality of a message can be compromised where the message can be seen by a man-in-the-middle by interrupting the communication in the middle without both victims realizing. Conti et al. presents the ARP spoofing for the man-in-the-middle to stay in the communication in silence as shown in Fig. 1. The victim, Bob would send a message to Alice without realizing that Eve is actually the one that sent a message to both victims. This would cause a confidentiality risk to the conversations between both victims even though they might believe that their conversation is safe.

Integrity: Message integrity can be compromised by the man-in-the-middle attacking the communication and modifying the message. This is one of the possible effects caused by session hijacking; for example, the attackers can interrupt the information and then modify the message to get session access so that the attacker can access the resources. These types of attacks usually occur during the authentication process and cause a threat to users since the layer of security is not secure. Session hijacking [12] is very dangerous for Internet banking especially since it contains sensitive data. During MITM attacks, traffic is usually interrupted, and a spoofed certification is given to the client to mimic the server.

Availability: There are a lot of prevention methods to ensure the security of the communication during a man-in-the-middle attack. Since the attack starts from the early stage, the victims might expose the authentication procedure to the attackers. One of the methods is to identify if the message is compromised by the attacker during authentication. The issue with a man-in-the-middle attack is that message modification shows the interruption. That is why there is a lot of cryptography methods that involve key sharing to ensure the message is not compromised.

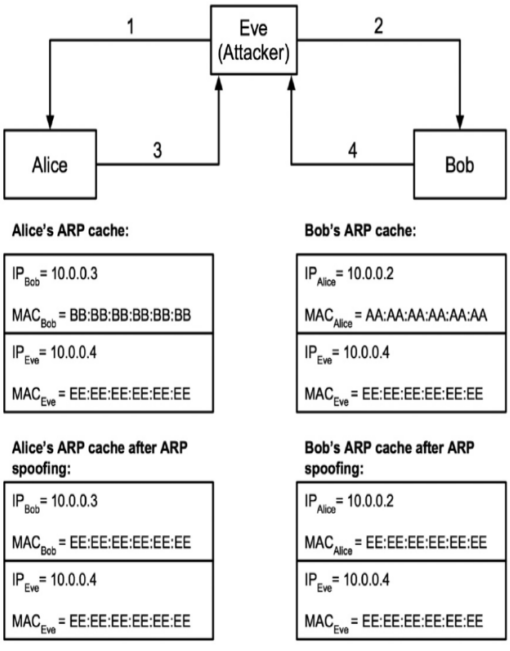


Fig. 1. Authorization Process in OAuth 2.0 [6].

### B. OAuth

OAuth is suitable to control the limited access of resources of the server.

For the server side, OAuth 2.0 is an authorization framework that is suitable for an environment that involves the use of multiple devices. OAuth 2.0 is an evolution from OAuth where it uses REST API as the development language. This specification is being developed within the Internet Engineering Task Force (IETF). The recent technology of OAuth 2.0 provides new security for users to enable third party applications to access resources from third party providers. These resources can be obtained using REST API[2]. For example, a person who already has a Facebook account can log in to the Spotify application through his Facebook account. Spotify does not have to know the username and password of the Facebook account.

The authorization process can be done using token authentication. Richardson and Ruby [6] present two types of authorization: using web interface or without web interfaces. Fig. 2 shows authorization with a web interface. In this example, the application needs permission from the user before Google provides a token to enable the application to use the Google calendar data.

For the client side, two types of platforms are considered, namely the Android apps and web application. For the Android apps to use third party libraries, an SDK [14] is normally provided. For example, an Android SDK is provided by Google for the programmer to develop apps without the need to register. This is different from using third party libraries online. For example, in a social network such as Facebook, Facebook SDK provides libraries for the Android platform. To access this SDK, a programmer has to create an account in Facebook and register as a programmer. The apps must be registered to Facebook and a secret key is provided for the apps to run online. The interaction is almost the same with web application. Facebook provides a system to enable programmers to register the application profile so that Facebook would recognize the list of applications that have access to the Facebook server.

Fig. 3 shows the sequence diagram of the authorization process. In this figure, a user refers to a person who has an account in the resource server, a user agent is the web application or Android app developed by the programmer. The client is provided by the middleware developer. Client act as interface. The user agent sends a request to the client, and the client will redirect the request to the URL of the server.

### C. Infrastructure-Less Communication

In an infrastructure-less communication, such as in a peer-to-peer or ad hoc network, there is no server to manage registered devices. The authorization process is done using a hello protocol. HELLO beacon messages per interval time in order to periodically update link information [15]. To follow hello beacon, the protocol would affect the network performances. Thus, to authorize users in infrastructure-less devices, a key provided by the programmer is sent by device A; if device B as the receiver returns the right string then device B is authorized.

[2] U. Friedrichsen ," OAuth 2.0: A standard is coming of age. codecentric AG". http://www.slideshare.net/ufried/oauth-20-18356495.

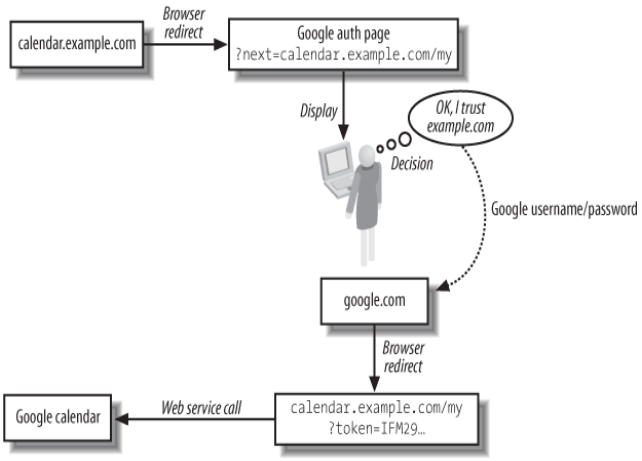


Fig. 2. Token to Access Functionality in Google Calendar [6].

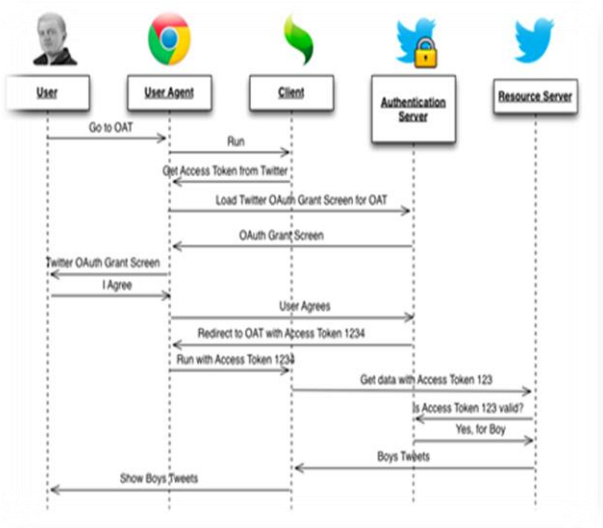


Fig. 3. Shows Authorization that Involves Third Party and user [14].

One of the solutions for security in peer-to-peer is provided by the Android API[3] is known as Elliptic Curve Diffie-Hellman [16] that sends a key without sharing the actual key. In this approach, the message is encrypted and decrypted. Cryptographic[3] hash functions are important security primitives especially for data integrity and authentication. HMAC-SHA1 is one of the cryptographic hash functions used to ensure data integrity in computing communication.

## III. Honeybee Computing Overview

Honeybee computing is a concept based on advanced ubiquitous computing to support Smart City Smart Village[1]. Honeybee computing contains several components such as semantic knowledge tool and predictive analytics for information management [17]. The sources of information in Honeybee computing are the web, public and private cloud, and devices.

In order to enable applications to be developed in a Honeybee computing environment, a middleware is needed. The architecture of the middleware to support the applications development in a Honeybee computing environment is described in [17]. The middleware architecture is shown in Fig. 4. There are five main components in the middleware, namely, Service Manager, Communication Manager, Security Manager, Semantic Manager, and News Manager. Network Management provides the connection with the user devices, while Resource Manager supports the management of the available resources. Honeybee computing supports two types of network: infrastructure or client-server based network and infrastructure-less or peer-to-peer network. In an infrastructure based network, users can communicate through a server. Communication in peer-to-peer network does not involve an intermediate server, but rather users communicate with each other directly. Most of the users access computing network through wireless communication either through Wi-Fi, ZigBee or 3G/4G technology. In a client-server based network, communication between the server and client is done through web services. There are two methods in providing web services: Representational State Transfer (REST) and Simple Object Access Protocol (SOAP). REST is a more popular choice for middleware as it is simpler to use.

Currently there are two types of applications: normal applications operating on a PC and apps operating on a mobile device. Development of apps within the Honeybee middleware is supported by a software development kit (SDK). Since a user may have a number of devices, the same application or app may be installed in some or all of these devices. Every application or app uses many different types of services. In order to control the type of services that can be accessed by an app, there is a need for permission control. There are two common types of permission provided by an application or app:

*1) Reading:* The application or app can obtain user's data stored in the honeybee server.

*2) Delete:* The application or app can delete user's data stored in the server.

Interaction between users, devices, applications and apps and permission is shown in Fig. 4.

The Honeybee Security manager is responsible in preventing malicious attacks to the server. The protection is done through an authorization key. Other components of the middleware are only accessible if the request contains the approved authorization key.

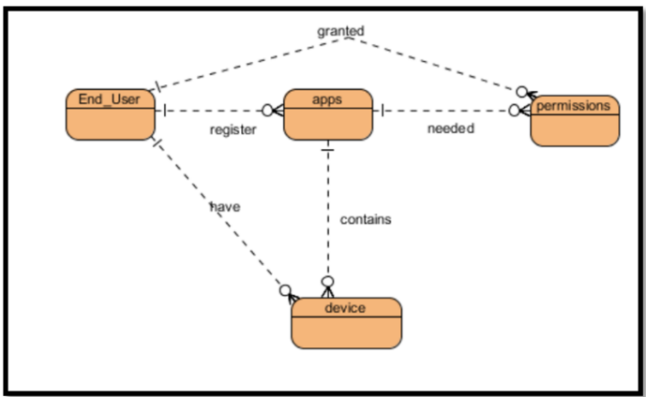


Fig. 4. Shows Interaction of the Security Package.

[3] N. Elenkov,"ECDH on Android sample app". Github. https://github.com/nelenkov/ecdh-kx.

Fig. 5 shows that there are two ways to access the services provided by the Honeybee Security manager. A Honeybee application on a PC can access the Security Web API directly. A Honeybee app on a mobile device will access the Security Web API through an SDK provided by org.honeybee.security.

There are two types of services provided by the Honeybee manager:

*1)* Authentication and authorization services to access services provided by other parts of the middleware; and

*2)* Encryption and decryption services for communication between apps on different devices.

Authentication is needed to access the Honeybee middleware. A server as a security manager provides service to ensure whether the app is authorized to access sources. For user sign up, login, and logout, a link to the Honeybee main web page is provided. After an authentication, a user has a session ID that is stored in cookies. The rules are as follows:

*1)* Programmer needs to register the apps to obtain information such as apps id and apps secret.

*2)* End_user is granted the type of permissions for the apps to access the source.

*3)* Each device provides a MAC address to access the server, and every token is provided for each device at each request to the server.

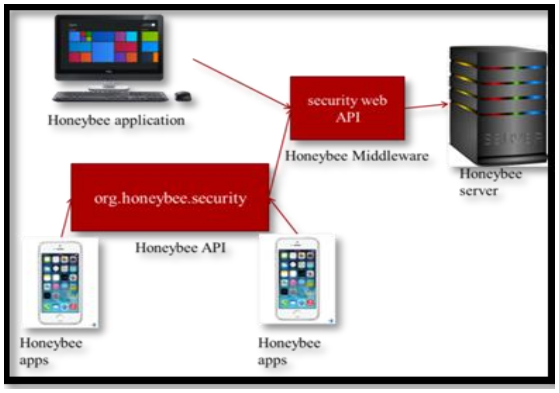


Fig. 5. Secret Generated by HMAC-SHA1 Algorithm for Programmer to Develop Application.

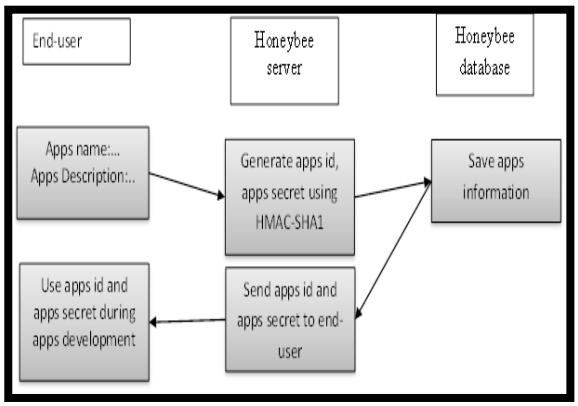


Fig. 6. Sequence Diagram in Security Manager.

In this system, a programmer is a user, thus user sign up is needed and then the user must register as a programmer to access the developer dashboard.

For an end-user to develop an app, registration is needed where the end-user needs to agree to the programmer agreement. Apps registration is needed for the Honeybee apps to access the Honeybee API. An app ID and app secret is provided to the end-user for development purposes; the sequence is shown in Fig. 6.

## IV. Authorization in Honeybee Computing

Honeybee computing authorization follows the OAuth workflow. The authorization is processed using multiple predefined URLs, called endpoints. There are 4 endpoints:

*1)* Request URI (this endpoint passes the request token).

*2)* Access URI (exchanges request token for an access token).

*3)* Authorize URI (confirms that the access token is valid).

*4)* Refresh token (refresh access token if previous is invalid).

Fig. 7 shows the sequence diagram between user, programmer, client, and server. The programmer develops the Honeybee app or web application and interacts with the Honeybee client before redirecting to the endpoints in the server. This process is adapted from the OAuth2.0 security.

Each number in the diagram is explained as follows:

- Request URI: The first endpoint is the request URI. This request URI is provided by the packages in Honeybee computing. The request URI passes the app secret to check if the user has granted permission to the Honeybee app to access the user account.
- Access URI: In the second endpoint, after the user is granted permission, an access token is provided. This access token can be used for requests to the resource server.
- Authorize URI: In this third endpoint, the access token is checked whether it is valid or not; if the token is valid then data is returned from the resource server.
- Refresh token URI: The fourth endpoint is used to refresh expired tokens; this is used when the user opens the app and the token saved is expired. This endpoint is not shown in the diagram. From Fig. 7, users must grant permission to enable third party applications to the access server.

The Honeybee computing Security manager is responsible for request management. This is to ensure that all requests are authorized. This is because every resource in Honeybee computing needs permission from the user and tokens to get authorized. The template is designed so that author affiliations are not repeated each time for multiple authors of the same affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization). This template was designed for two affiliations.

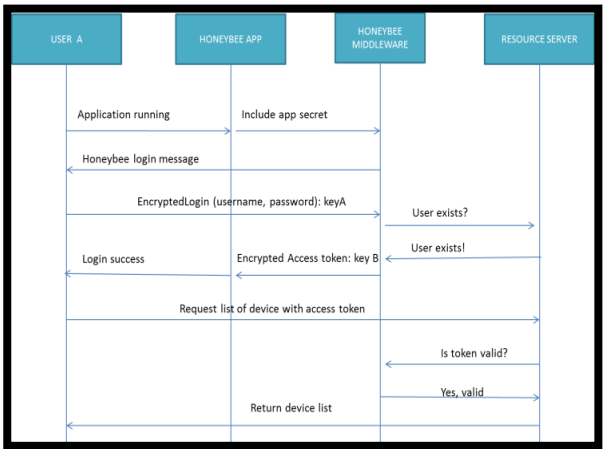


Fig. 7. Communication Flow when Device uses WAP Connection.

## V. Communication in Honeybee Computing

There are a number of communication processes involved in Honeybee computing: user registration, application registration, accessing server for resources, finding nearby devices, communicate with other devices, peer-to-peer communication and building an ad hoc network. The descriptions of these processes are as follows:

User registration: Before a user can use the application, the user must have an account with the Honeybee system. This registration is different from the application registration; user must grant different types of permissions. This procedure is to protect multiple access to user data. In every user device there are a lot of applications, however only applications that are granted by the user can access the Honeybee server.

Application registration: Since the Honeybee system is a community system, the third party application would need to be authenticated by the Honeybee server. Thus, the app developer or programmer must register the application to the system. Programmers have to input the application package and then the app ID and app secret are provided for that particular application as a signature to access the server.

Accessing server for resources: The sequence of activities to assess a Honeybee resource is shown in Fig. 8. When a user is successfully logged in to the system, the middleware will provide a token and the application can request the resource by using the provided token. Fig. 8 shows a device communication when connected to WAP.

Finding Nearby Devices: To find a nearby device, the user must use Wi-Fi to connect to the Honeybee server. User must log in to the Honeybee system to get information needed from the server. Honeybee SDK then lists all nearby users who are online using WAP.

Communicate with other devices: The steps needed for communicating with other mobile devices using Wi-Fi are as follows:

*1)* Device A sends an encrypted message and key to device B. Device B then replies with an encrypted message and key.

*2)* Device A then decrypts the message and checks the following:

*a)* Decrypted message follows the format of the message.

*b)* MAC address inside the message exists in the file downloaded from the Honeybee server.

Device B would follow the same procedure to identify if the key is modified. During this step, both devices can proceed using the key or stop the session and start with a new session to get a new key.

*3)* If both devices agree to proceed, device A sends an encrypted message to device B without the key sending new keys.

Peer-to-peer communication: Peer-to-peer communication uses a concept similar as WAP since both use Wi-Fi to communicate; at the same time peer-to-peer is also an ad hoc network. To explain the communication, the steps of the procedure are as follows:

*1)* Device A scans for nearby devices using the Android library and identifies which device to send a message. Device A then encrypts the message and sends with a key to the selected device, which is device B. Device B then replies with the encrypted message and key by following the format needed.

*2)* Device A and device B then decrypt the message given and identify the following conditions:

*a)* Decrypted message follows the format to send message; and

*b)* One of the attributes is following the file downloaded from the Honeybee server earlier.

If any of the conditions are not followed, the communication is suspected as not secure. Both devices could proceed or stop the current session. If both devices agree to continue, the message can be sent by sending an encrypted message without the need to send the key.

Building an ad hoc network: An ad hoc network involves a number of devices connected together to form a network. It does not involve an intermediate server to send messages from one device to another, thus the communication is real time. By using the same authentication method as WAP, the app then can communicate. After logging in to the Honeybee system, the system would provide the information of registered devices to the Honeybee system.

## VI. Conclusion

In man-in-the-middle attacks, there are multiple methods to prevent these attacks. There are several methods to prevent the effect of the attacks such as focus on the authentication method and using a different technique of cryptography for messages. There are also methods that focus on the digital signature to prevent any attacks. Honeybee computing security on the other hand, adapts the cryptography method and also validates the application at the server. To get the list of registered devices, the device communicates with the server before connecting to the device. These two communication need to identify if a man-in-the-middle attack is possible during data transfer. To prevent the man-in-the-middle attack, three methods of prevention are covered which are identity identification, resource protection and communication secure.

To ensure the authorization, the OAuth 2.0 method is used. This type of authorization using key sharing involves user agreement before proceeding. With this method, not all data is accessible to the user. This method is also located at the server the location of the resources. Other than that, every authorization and authentication process involves more than one secret key, thus the process to ensure the security of the communication is complex to get easily attacked by intruders. Since the man-in-the-middle is capable of impersonating the victim, one of the approaches in Honeybee computing is identity identification. To ensure if the sender is not an attacker, the server provides a list of devices that helps the user to identify if the sender is actually a registered user or an attacker. Since the list of registered devices is downloaded from the server, the attacker would need to attack the communication with the server to modify the list. By using the OAuth 2.0 method that is secure for the environment, the attacker would have difficulties to interrupt the communication.

One of the causes of man-in-the-middle is message availability where the attacker changes the message without victims realizing. For this case, we use cryptographic and also adopt the key sharing method. To prevent the key from being attacked by intruders, we use the ECDH key for key sharing. This method is one of the methods that are used for man-in-the-middle attacks to prevent any modification to the message. If the attacker is capable of accessing the message, the communication is still secure since the attacker would face other difficulties in decrypting the message. The communication design in Honeybee computing is mainly based on the entities inside the system. Two types of users which are application user and programmer show that Honeybee computing is a community system. Since there is involvement of third party applications in accessing the resources, the security involvement from different aspects is very important. With multiple devices that use multiple applications to access the resources, the OAuth 2.0 technique is used. Since Honeybee computing architecture involves the infrastructure-less communication, the ECDH method is adopted in the Honeybee computing security. These two methods prevent man-in-the-middle attacks to the system. The effects of man-in-the-middle attacks would cause a lot of problems.

As conclusion, it is important for any software that involved with communication transmission to have a secure data transmission. The after effect of MITM would cause many troubles in the resources To ensure the proposed authorization and authentication help to secure communication in honeybee computing, the future research is needed, which is to test whether the communication is secure from MITM attack, a simulation is needed where MITM attack to the application developed in the honeybee computing. With the functionality provided in honeybee computing such as send message, request data from middleware, and register user info, all the functionality must be tested to ensure MITM attack would not happen.

The next task is to validate the proposed security mechanism. The validation process will be done using security testing based on the test cases [18].

## ACKNOWLEDGMENT

The authors would like to thank Mr. Muhammad Zharif for his valuable suggestions on the improvement of the paper. This research is one of the Malaysian Government funded projects under the Ministry of Higher Education (MOHE) Fundamental Research Grant (FRGS) with reference number FRGS/1/2015/ICT04/UKM/02/3.

## REFERENCES

[1] K. Salamzada,Z. Shukur and M. Abu Bakar , “A Framework for Cybersecurity Strategy for Developing Countries: Case Study of Afghanistan”. Asia-Pacific Journal of Information Technology and Multimedia,4(1): 1 – 10.

[2] Sidra Ijaz, Munam Ali Shah, Abid Khan and Mansoor Ahmed, “Smart Cities: A Survey on Security Concerns” International Journal of Advanced Computer Science and Applications(IJACSA), 7(2), 2016.

[3] A. Kargar Raeespour A and AM Patel, “Design and Evaluation of a Virtual Private Network Architecture for Collaborating Specialist Users”. Asia-Pacific Journal of Information Technology, 5 (1):15 – 30.

[4] MA Catur Bhakti and A. Abdullah, ”EAP Authentication Mechanism for Ad Hoc Wireless LAN”, Journal of Information Technology and Multimedia, 5(2008): 13-40.

[5] V. Rastogi and A. Agrawal, “All your Google and Facebook logins are belong to us: A case for single sign-off”. 2015 Eighth International Conference on Contemporary Computing (IC3) (2015), Noida, 20th–22nd August, India.

[6] L. Richardson and S Ruby, “RESTful web service”. 1st ed. O’Reilly Media.

[7] Mohammed Nasser Al-Mhiqani, Rabiah Ahmad, Warusia Yassin, Aslinda Hassan, Zaheera Zainal Abidin, Nabeel Salih Ali and Karrar Hameed Abdulkareem, “Cyber-Security Incidents: A Review Cases in Cyber-Physical Systems” International Journal of Advanced Computer Science and Applications(IJACSA), 9(1), 2018.

[8] T. Ziebermayr and S. Probst, “Web Service Authorization Framework”. Proc. ICWS, 614-621.

[9] H. Lu , “Keeping Your API Keys in a Safe”. Proc. CLOUD, 962-965.

[10] J. Franks, P. Hallam-Baker, J. Hostetler,S. Lawrence,P. Leach,A. Luotonen and L. Stewart, “RFC 2617: HTTP Authentication: Basic and Digest Access Authentication”. IETF. https://tools.ietf.org/html/rfc2617#section-2

[11] Muhammad Kazim and Shao Ying Zhu, “A survey on top security threats in cloud computing” International Journal of Advanced Computer Science and Applications(IJACSA), 6(3), 2015.

[12] M. Conti, N.Dragoni, and V. Lesyk, “A Survey on Man in the Middle Attack”. IEEE Communications Surveys and Tutorials, 18(3):2027-2051.

[13] AA. Maksutov,IA. Cherepanov and MS. Alekseev, “Detection and prevention of DNS spoofing attacks”. 2017 Siberian Symposium on Data Science and Engineering (SSDSE).

[14] T. Ziebermayr and S. Probst, “Web Service Authorization Framework”. Proc. ICWS, 614-621.

[15] E. Khan, M. El-Kharashi, F. Gebali and M. Abd-El-Barr,”Design space exploration of a reconfigurable HMAC-hash unit”. Journal of Research and Practice in Information Technology. 40(2):109.

[16] E. Khan,M. El-Kharashi,F. Gebali and M. Abd-El-Barr ,” Design space exploration of a reconfigurable HMAC-hash unit”. Journal of Research and Practice in Information Technology. 40(2):109.

[17] NH. Azizul, A. Mohd Zin and E. Sundararajan,”The Design and Implementation of Middleware for Application Development within Honeybee Computing Environment”. IJASEIT, (6)6:937-943.

[18] A. Lunkeit and I. Schieferdecker “Model-Based Security Testing - Deriving Test Models from Artefacts of Security Engineering”. 2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 244-251.