An efficient model for load balancing and communication security in a Multi Agent System Middleware based on Json Web Token

Badr Eddine Sabir[†] **Mohamed Youssfi**[††] **Omar Bouattane**[†††] **and Hakim Allali**[††††]**,**

[†, †††] Laboratory of Watch for Emergent Technologies FST, Hassan I University of Settat, Morocco
[††, ††††] Laboratory SSDIA, ENSET Mohammedia, Hassan II University of Casablanca, Morocco

Summary
In this paper, we propose a new model based on Json Web Token (JWT) for load balancing and communication security as part of a Multi Agent System Middleware for massively distributed systems. The proposed model aims to provide better load balancing and secure communications between agents to ensure the integrity of the exchanged messages, the authentication of agents, and the no-repudiation. This architecture is based on the Stateless JWT security technology based on asymmetric cryptographic algorithm used for validation of subsequent client requests for making frequent remote calls to the target server resources. The proposed solution uses a digital signature claim to ensure the integrity of the message, the authentication of the emitter and the non-repudiation based on the asymmetric cryptographic technology. The article presents a conceptual approach based on digital trust micro-agent for better security and load balancing management.

Key words:
Authentication, Digital signature, Multi-Agent Systems, Middleware, Json Web Token, Load balancing.

1. Introduction

In recent years, we have testified an impressive evolution of information technologies Data-intensive and compute-intensive applications require huge computing power and very large data storage resources. Traditionally, to speed up calculations, massively parallel machines are used, composed of a multitude of microprocessors interconnected with each other according to different topologies. This is the case of the GPU architectures used to handle an extensive amount of computation [1][2][3][4][5], which is widely used in recent years to accelerate graphics applications as well as other parallel applications with large mass of data on various scientific fields [6] [7].
However, a single parallel machine, as powerful as it is, is not enough to cope the explosion of masses of data that applications must analyze and process in a very short time. Distributed systems used by large-scale systems [8] are therefore an essential and a necessary solution for High-Performance Computing (HPC) [9].

Manuscript received April 5, 2019
Manuscript revised April 20, 2019

In fact, with the evolution of computer networks performance and the development of new high-performance middleware distributed systems, it is possible to collaborate a multitude of heterogeneous computers to achieve high performance for compute-intensive or data-intensive applications.
The Multi-Agent Systems (MAS) present themselves as an attractive solution for modeling complex distributed systems. In the field of MAS, some of the MAS Middleware are widely used to implement models based on Multi Agent approaches. However, in applications using a very large number of agents, these Middlewares have shown many limits in terms of performance as well as in terms of communication security between the agents.
Therefore, these two fundamental questions must be answered:

- How to ensure the security of the communication between the agents?
- How to ensure load balancing, fault tolerance, and scaling, which are decisive factors in determining the reliability of such a system?

In this paper, we present a conceptual approach to the proposed architecture describing a flexing, lightweight, and secure system for better security and load balancing management in Multi-Agents System, answering the two questions above, based on a digital trust agent.
After presenting the related works in the second section, the third section is devoted to the description of the JSON Web Token (JTW). In the fourth section, we will present the proposed model and its description. The last section gives some concluding remarks and perspectives for the future related works.

2. Related Work

2.1 Load Balancing

In reference [10] the authors present a new model built around a Multi-Agent System middleware platform for massively distribution and execution of heterogeneous and independent tasks in a heterogeneous distributed system. The platform allows the concurrent execution of several different types of tasks coming from different nodes. To uniformly perform the tasks, the proposed model is based on an abstraction layer which represents separately the task, its metadata and its data using the object-oriented approach. The platform contains five modules allowing for each task, its configuration, production, assignment, execution, and its following-up. The main module of this platform is the assignment one that uses an efficient load balancing procedure based on aspect-oriented approach. This assignment procedure uses the task metadata, the communication latency and nodes performance to associate the appropriate executor agent to the producer one. The model presents a deployment and caching system of the task code to strongly reduce the amount of data exchanged between the agents of the platform. These properties obviously prove the efficiency of the proposed model.

In reference [11], authors propose a framework of parallel remote sensing image interpretation in desktop grid. This framework consists of modules such as job partitioning and scheduling, load balancing, communication, and image processing.

In reference [12], the authors present a new distributed computing middleware for High Performance Computing (HPC) based cloud micro-services. Besides, the proposed middleware implements a new cooperative micro-services team works model for massively parallel and distributed computing. This model is constituted by distributed micro-services as Microservice Virtual Processing Units (MsVPUs).

This middleware integrates the load balancing mechanism that ensures the synchronization between the computing components. Additionally, it implements light-weight communication mechanism using the asynchronous communication protocol AMQP. Also, it implements load balancing strategies that handle micro-services and ensures that the HPC parallel programs are distributed by hinging on a micro-service. This micro-service is the one responsible for the management of the micro-services in the distributed system. It can easily get the micro-services nodes performance to load balance the applications request between heterogeneous nodes according to a load balancing strategy.

Several load balancing algorithms, designed for distributed systems, have been proposed and reviewed [13] [14].

2.2 Multi-Agent System Security

In reference [15], authors proposed a security policy responding to security properties (access control, authentication, confidentiality, integrity) based on a normative Multi-Agent System for securing complex applications. In the complex systems, the information flows are dense and confused consequently, risks and menaces rates increase more and more. This model considering the high heterogeneity of the agent execution environments in terms of resources and computing devices and the high dynamicity that causes frequent changes in the agent execution context. Also, they have focused on the notion of context which becomes more and more convincing the Multi-Agent's System community by its importance to respond to novel challenges. In this perspective, they have used the notion of norms to specify the secured behaviour of agents in such environments (norms manage a variety of agents and regulate a uniform behaviour within a social group), adopting the ontology for the specification of contexts. In fact, it contributes in term of expressivity and subjectivity and averts ambiguity. The Model is high-lighted by an architecture showing the distribution of the Multi-Agent System in the environment.

In reference [16], authors propose to secure architecture of mobile cloud computing composed of a cluster of mobile devices suffering from some security problems such as user's privacy when different node communicate with each other, by integrating a Multi-Agent System which provides the required intelligence level to overcome privacy, availability issues and to support the computing performance. This new architecture based on the concept of the MAS provides a high level of security during the data transfer on the mobile network and to optimize the computing. JADE platform has been used as Multi-Agent System platform, allowing to encrypt the data between nodes and to split and assign the processing to a different mobile device which allows to speed up the processing and therefore increase the performance and dwindle the energy cost of the mobile device while preserving privacy and integrity of user's data.

In [17], authors propose a lightweight security protocol for integrated agents and smart objects that facilitate cooperation and global intelligence, they have implemented and enhanced a secure mobile agent protocol, Broadcast based Secure Mobile Agent Protocol (BROSMAP), to enable interoperability and global intelligence with smart objects in the Internet of Things with consideration of overall performance, using Elliptic Curve Cryptography (ECC) which is one of the most secure protocols that provides confidentiality, authentication, authorization, accountability, integrity, and non-repudiation. The authors have conducted an experimental performance evaluation to compare and measure the performance of RSA-based BROSMAP, and ECC-based BROSMAP on a client side (Android Java) and server side (XAMPP). The proposed lightweight security protocol for Multi-Agent based IoT systems was verified using Scyther tool before the implementation process. They have measured the performance in terms of important metrics which include execution time and computational cost. They have also compared performance measures on varying key sizes. In terms of execution time, concluding that the ECC is almost twice as fast as RSA 2048 and 4 times faster than RSA 3072. The main reason behind the superior performance of ECC-BROSMAP is the reliance on smaller key sizes and encrypting/decrypting using symmetric cryptography only. The authors recommend for systems with limited resources such as IoT devices and agent-based systems that require high security, the use of ECC-BROSMAP over RSABROSMAP. ECC-BROSMAP provides all the security requirements that RSA-BROSMAP provides but it is more efficient and lightweight because of the elimination of asymmetric encryption, the use of smaller key sizes and the combination of ECC keys with symmetric encryption only.

In [18], authors offered a dynamic federated identity management model that allows rapidly decrease and increase the scope of the cloud. For this reason, they applied the MAS allowing to perform the collection of necessary information about the user, which allows making dynamic decisions in the real-time system. For providing a user's single access to cloud infrastructure the protocol SAML is used. The users' dynamic federated identity management process is based on PKI technology and the calculation of trust value. The user's access to the cloud infrastructure is provided based on their attribute certificates, also the privileges to each user are given by the role based on the access control mechanisms. Cloud environments are dynamic, because of that statically the determination of the users' roles only by certificates has lost its actuality. In this case, the dynamic management of roles becomes necessary. The application of multiagent systems in architecture allows the dynamic determination of users' roles. The content of these roles also contains behavioural characteristics of the users. Multi-Agent System provides the authentication, authorization and audit operations within the cloud environment. The trust degree of the MAS module is defined by the Public Key Infrastructure (PKI).

2.3 Deduction

After this first part we concluded that:

- The solutions proposed to secure communication between agents, namely integrity, authentication, and non-repudiation, adds a layer of complexity in the development of these solutions in addition to presenting a risk hinder the performance of this middleware.
- Also, the work presented to ensure load balancing is still not simple and obvious to implement, especially taking into consideration the fault tolerance.

3. Background

JWT is a compact, URL-safe means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JSON object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web (JWE) structure, enabling the claims to be digitally signed or integrity protected with a Message Authentication Code (MAC) and/or encrypted [19]. JWTs are based on Web JWS and JWE [20]. it consists of three structures separated by a full stop (.), namely:

- Header. There are two parts in the header, one of them is token type, namely JWT, and another one is hashing algorithm used, such as HMAC SHA256 or RSA.
- Payload. It is the second part of token which consists of claim. Claim is the statement about an entity (usually users) and supplementary metadata.
- Signature. To create a signature, JWT must follow the JWS specification. JWS is the content secured by digital sign or MACs which uses JSON as the basic data structure [21].

JWT access tokens can be used for validation of subsequent client request without making frequent calls to the resource server or database. Access tokens can have limited validity periods via embedded expiration time. Access-related claims can also be embedded as part of its payload [22].

4. Proposed model

4.1 Components of the proposed model

The proposed model, using JWT, makes it possible to secure the communication between the agents by ensuring the integrity of the exchanged messages, the authentication of the sender and the non-repudiation, based on the asymmetric encryption algorithms. It also facilitates the implementation of a load balancing management solution by managing the sessions by the agents themselves without calling on third-party applications, which ensures better performance and better management of the fault tolerance.

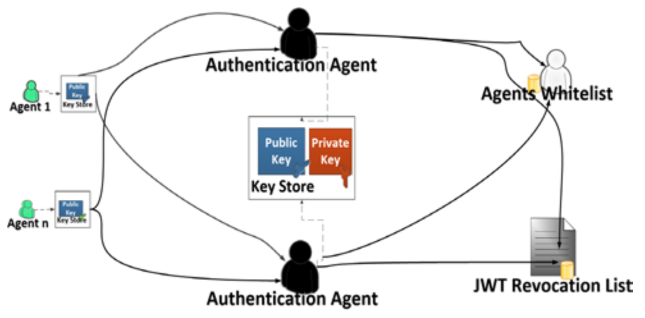


Fig. 1 Proposed model architecture.

- Agents 1…n: Agents belonging to the domain to which the Authentication Agent belongs (Machine, logical domain, desktop grid, ...).
- Authentication Agent: Responsible for the generation of JWT for the different agents wants to communicate with other agents. The JWTs generated by this Agent are used to ensure the integrity, authentication and non-repudiation of the exchanges between Agents. Several Authentication Agents can ensure the generation of JWT using the same private key to ensure fault tolerance and avoid the Single Point of Failure.
- Agents Whitelist: Responsible of adding the newly created agents and belonging to the same domain of the Authentication Agent in the whitelist and checking the membership of the agents in this domain. Before generating a JWT for an Agent, the Authentication Agent must first ensure that the Agent requesting the JWT belongs to the Authentication Agent domain. The whitelist agent performs this function thanks to its whitelist.
- JWT Revocation List (JRL): is a list of JWTs that has been revoked by an Authentication Agent before their scheduled expiration date and should no longer be trusted. JRLs are a type of blacklist and are used by all Authentication Agents to verify whether a JWT is valid and trustworthy. When an Authentication Agent must trust an Agent X, he checks that the JWT is not listed in a JRL. These checks are crucial steps in any transaction agents because they allow to verify that the JWT still trustworthy.
- KeyStore
    - o Private Key: Used by the Authentication Agents for signing generated JWTs.
    - o Public Key: Used for the verification of all JWTs generated by the Authentication Agent.

4.2 Use Case diagram

The following schema shows the Use Case diagram of the proposed model:

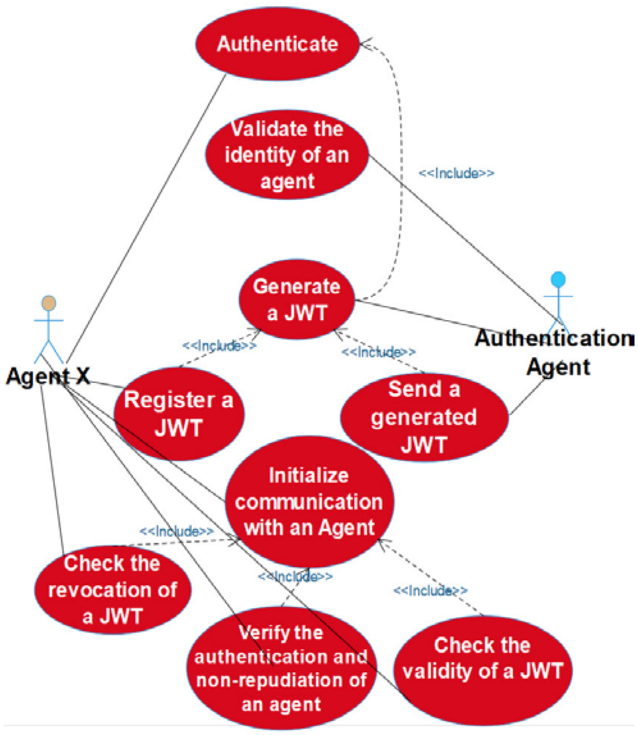
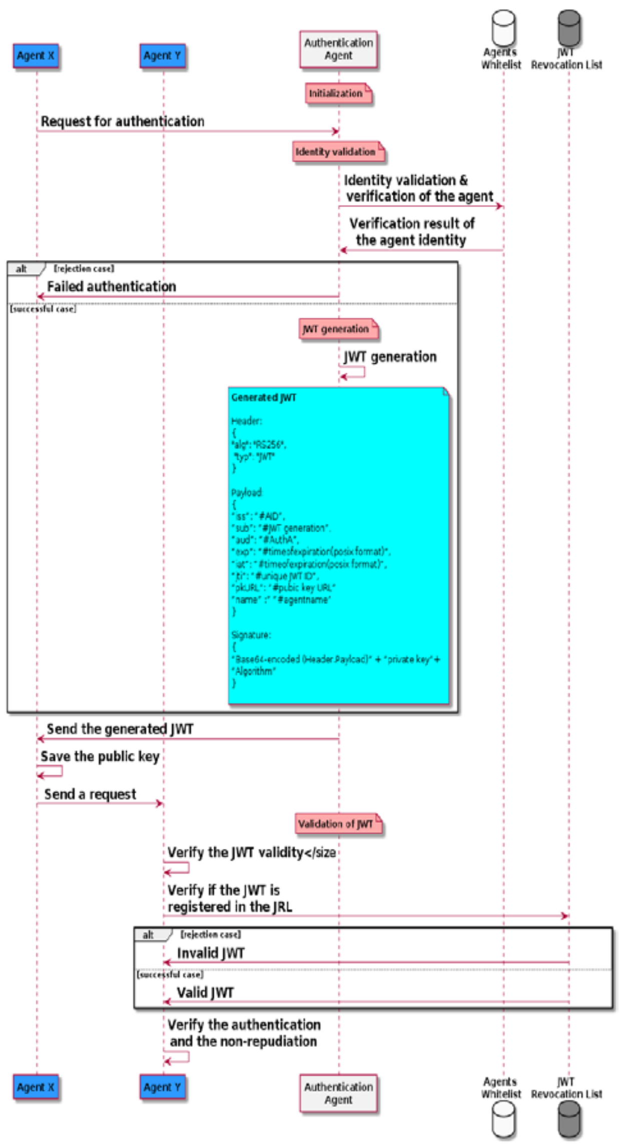

Authenticate
Validate the identity of an agent
<<Include>>
Generate a JWT
Authentication Agent
Agent X
<<Include>>
<<Include>>
Register a JWT
Send a generated JWT
Initialize communication with an Agent
<<Include>>
Check the revocation of a JWT
<<Include>>
<<Include>>
Verify the authentication and non-repudiation of an agent
Check the validity of a JWT

Fig. 2 Synthesized Use Case diagram of the proposed model.

4.3 Generation of a new JWT

The following diagram shows the steps of generating a JWT, checking the validity of the JWT and verifying the authentication and non-repudiation of agents.



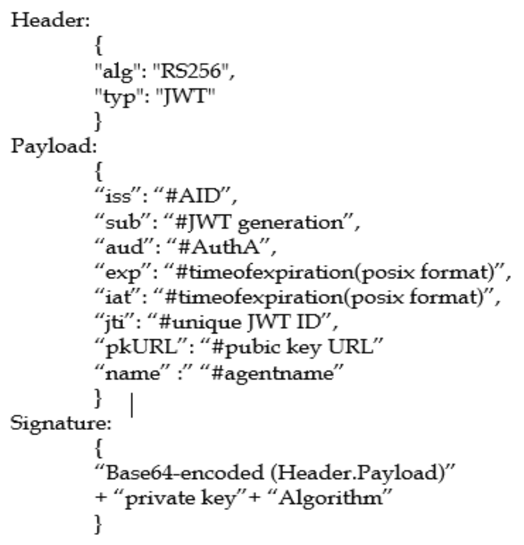
```
Generated JWT

Header:
{
"alg": "RS256",
 "typ": "JWT"
}

Payload
{
"iss" : "#AID",
"sub": "#JWT generation",
"aud": "#AuthA",
"exp": "#timeofexpiration(posix format)",
"iat" : "#timeofexpiration(posix format)",
"jti" : "#unique JWT ID",
"pkURL": "#pubic key URL"
"name" :" "#agentname"
}

Signature:
{
"Base64-encoded (Header.Payload)" + "private key" +
"Algorithm"
}
```

Fig. 3 New JWT generation scenario.

If you would like to itemize some parts of your manuscript, please make use of the specified style "itemize" from the drop-down menu of style categories
Here are the steps of generating a new JWT in chronological order:

- Agent X requests the Authentication Agent to generate a JWT for it to communicate with Agent Y;
- The Authentication Agent verifies if Agent X belongs to the Whitelist by checking if his Agent ID (AID) is registered in this list;
- The Whitelist Agent returns the response to the Authentication Agent;
- If Agent X belongs to the Whitelist, then: The Authentication Agent generates a JWT for Agent X using its private key based on the following equation:

$$f(base64Encoded)(header.payload.signature) \quad (1)$$



```
Header:
        {
        "alg": "RS256",
        "typ": "JWT"
        }
Payload:
        {
        "iss": "#AID",
        "sub": "#JWT generation",
        "aud": "#AuthA",
        "exp": "#timeofexpiration(posix format)",
        "iat": "#timeofexpiration(posix format)",
        "jti": "#unique JWT ID",
        "pkURL": "#pubic key URL"
        "name" :" "#agentname"
        }  |
Signature:
        {
        "Base64-encoded (Header.Payload)"
        + "private key"+ "Algorithm"
        }
```

Fig. 4 Format of the generated JWT.

  Else, the Authentication Agent returns a message to Agent X indicating that his request is rejected;
- The Authentication Agent sends the generated JWT to Agent X. The generated JWT contains the URL to retrieve the public key that must be used for the verification of all JWTs generated by the Authentication Agent;
- Agent X retrieves the JWT, then the public key via the URL indicated in this JWT to stores it in its own Key-Store;
- Agent X tries to prove his identity to Agent Y by inserting the recovered JWT into the send message;
- Agent Y checks the validity of the JWT contained in the Agent X request. The verification operation is performed by carrying the following operations:
  - o Retrieving the public key of the Authentication Agent certificate that is in his KeyStore;
  - o Comparison of the recovered public key with that deposited by the Authentication Agent:

- If it finds that the public key has not changed, it uses the one stored on his KeyStore.
- If not, it gets the new public key, puts it in his KeyStore, then carries out the verification.

- Agent Y after checking the validity of the JWT, checks if the JWT is not registered in the JRL:
  - If the JWT is on this list, it rejects the request of Agent X.
  - Else it considers this request.
- At this level, the Agent Y is sure that the message has not been tampered with and the JWT is valid. To ensure authentication and non-repudiation, Agent Y can retrieve the Agent AID contained in the JWT and then compare it with AID of Agent X, and if both AIDs are the same that means that the JWT belongs to Agent X.

5. Conclusion

An architecture based on the JWT technology has been proposed in this document to ensure a better load balancing on one hand, and a secure channel for the exchanged messages between agents based on a digital trust agent on the other hand.
The model takes advantage of the electronic signature of JWTs to ensure the integrity of the messages exchanged between agents, the authentication of agents, and non-repudiation, based on the asymmetric cryptographic technology.
In addition, through JWT's stateless functionality this architecture provides a load balancing that is efficient and easy to implement.

The simulated tests and practical implementation of the proposed model will be a cause for future research to measure stability and performance in case of scalability and possible safety failures.

References

[1] H. Lee and M. A. Al Faruque, "GPU Architecture Aware Instruction Scheduling for Improving Soft-Error Reliability," in IEEE Transactions on Multi-Scale Computing Systems, vol. 3, no. 2, pp. 86-99, April-June 1 2017.

[2] [2] H. Lee, M. A. A. Faruque, "GPU-EvR: Run-time event based real-time scheduling framework on GPGPU platform", Proc. Des. Autom. Test Europe Conf. Exhibition, pp. 1-6, 2014.

[3] A. Maghazeh, U. D. Bordoloi, A. Horga, P. Eles, Z. Peng, "Saving energy eithout defying deadlines on mobile GPU-based heterogeneous systems", Proc. Int. Conf. Hardware/Softw. Codesign Syst. Synthesis, pp. 1-10, 2014.

[4] G. Sadowski, "Design challenges facing CPU-GPU-Accelerator integrated heterogeneous systems", Proc. 51st Des. Autom. Conf., pp. 1, 2014.

[5] Z. Chen and D. Kaeli, "Balancing Scalar and Vector Execution on GPU Architectures," 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Chicago, IL, 2016, pp. 973-982.

[6] Z. Juhasz, "Highly parallel online bioelectrical signal processing on GPU architecture," 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, 2017, pp. 340-346.

[7] G. Zhao, Y. Liu, S. Zhang, F. Shen, Y. Lin and G. Shi, "Parallel Implementation of the Range-Doppler Radar Processing on a GPU Architecture," 2016 15th International Symposium on Parallel and Distributed Computing (ISPDC), Fuzhou, 2016, pp. 76-79.

[8] [G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of V. Kulba and S. Somov, "Problem of optimal placement of data files in large-scale unreliable distributed systems," 2017 Tenth International Conference Management of Large-Scale System Development (MLSD), Moscow, 2017, pp. 1-5.

[9] I. A. Jothi and P. Indumathy, "Increasing performance of parallel and distributed systems in high performance computing using weight based approach," 2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015], Nagercoil, 2015, pp. 1-4.

[10] [A. Daaif, O. Bouattane and M. Youssfi, "An efficient multi-agent computationnal model for massively distribution of independent and heterogeneous tasks," 2017 Intelligent Systems and Computer Vision (ISCV), Fez, 2017, pp. 1-7.

[11] Dongsheng Li, Jiannong Cao, Xicheng Lu, Keith C. C. Chen, "Efficient Range Query Processing in Peer-to-Peer Systems," IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 1, pp. 78-91, Jan. 2009, doi:10.1109/TKDE.2008.99.

[12] F. Z. Benchara, M. Youssfi, O. Bouattane and H. Ouajji, "A new efficient distributed computing middleware based on cloud m[icro-services for HPC," 2016 5th International Conference on Multimedia Computing and Systems (ICMCS), Marrakech, 2016, pp. 354-359.

[13] D.R. Karger and M. Ruhl, "Simple Efficient Load-Balancing Algorithms for Peer-to-Peer Systems," Theory of Computing Systems, vol. 39, pp. 787-804, Nov. 2006.

[14] Ioannis Konstantinou, Dimitrios Tsoumakos, Nectarios Koziris, "Fast and Cost-Effective Online Load-Balancing in Distributed Range-Queriable Systems," IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 8, pp. 1350-1364, Aug. 2011, doi:10.1109/TPDS.2010.200.

[15] H. Cheribi and M. K. Kholladi, "A security model for complex applications based on normative multi-agents system," 2015 Second International Conference on Information Security and Cyber Forensics (InfoSec), Cape Town, 2015, pp. 41-46.

[16] [16] A. Abdellaoui, A. Laksantini and H. Chaoui, "A security scheme for mobile cloud using multi-agents system," 2016 4th IEEE International Colloquium on

Information Science and Technology (CiSt), Tangier, 2016, pp. 615-620.
[17] H. Hasan et al., "Secure lightweight ECC-based protocol for multi-agent IoT systems," 2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Rome, 2017, pp. 1-8.
[18] R. M. Alguliev and F. C. Abdullayeva, "Identity management based security architecture of cloud computing on Multi-Agent Systems," Third International Conference on Innovative Computing Technology (INTECH 2013), London, 2013, pp. 123-126.
[19] Michael B. Jones, John Bradley, and Nat Sakimura. JSON Web Token (JWT).2015. https://tools.ietf.org/html/rfc7519.
[20] P. Solapurkar, "Building secure healthcare services using OAuth 2.0 and JSON web token in IOT cloud scenario," 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), Noida, 2016, pp. 99-104.
[21] M. Jones, J. Bradley, N. Sakimura, “JSON Web Token (JWT)”. IETF, May 2015.
[22] O. Ethelbert, F. F. Moghaddam, P. Wieder and R. Yahyapour, "A JSON Token-Based Authentication and Access Management Schema for Cloud SaaS Applications," 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud), Prague, 2017, pp. 47-53