A Novel Design of Authentication-as-a-Services (AaaS) Architecture in Cloud Computing

Yu-Yi Chen[1*]   Jun-Chao Lu[2]   Jinn-Ke Jan[2]

[1] Department of Management Information Systems, National Chung-Hsing University
Taichung 402, Taiwan
chenyuyi@nchu.edu.tw

[2] Department of Computer Science and Engineering, National Chung-Hsing University
Taichung 402, Taiwan
{phd9416, jkjan}@cs.nchu.edu.tw

Received 20 February 2013; Revised 18 June 2013; Accepted 20 June 2013

Abstract. Today, there are various cloud computing services. With only a few clicks, users can get the on-demand resources and services over internet without regard to where the services are hosted or how they are delivered. Moreover, the computing will become a kind of services that similar to traditional utilities such as water, gas, electricity, and telephony in the near-future. However, there are some security issue in Cloud computing should be solved. In this paper, we propose authentication-as-a-services (AaaS) architecture to provide a trust authentication infrastructure. In this design, user's sensitive information will not leak to the service provider. Moreover, user can roaming in various service providers without re-registration and re-authenticate. This design is a foundation for meshing-up a trust and anonymous Cloud computing.

Keywords: cloud computing; software services; privacy; authentication

1 Introduction

The computing will become a kind of services that similar to traditional utilities such as water, gas, electricity, and telephony in the near-future. Due to the increase in the technological advancements, this kind of "everything-as-a-service" (XaaS) [1] or called "cloud computing" [2] is becoming the hottest topics. The name cloud computing was inspired by the cloud symbol that is often used to represent the Internet in flow charts and diagrams. It is an innovative Information System (IS) architecture, visualized as what may be the future of computing, a driving force demanding of its audience to rethink their understanding of operating systems, client–server architectures, and browsers. Cloud computing results from the convergence of Grid Computing, Utility Computing, SaaS, and essentially represents the increasing trend towards the external deployment of IT resources. It has leveraged users from hardware requirements, while reducing overall client side requirements and complexity. With only a few clicks, users can get the on-demand resources and services over internet without regard to where the services are hosted or how they are delivered. Thus, the computing world is transforming towards developing software as a service, rather than to run on their individual computers. More and more services are offered in this way. Such as email, external data storage, and personal productivity applications offered by Google [3]; the social networking service offered by Facebook [4]; to the latest cloud rendering technology RealityServer [5] offered by NVIDIA ; and other cloud computing services [6-10].

As those enterprises toward providing Platform as a Service (PaaS) and Software as a Service (SaaS) for customer, the numbers of cloud platform are increasing. Kinds of Web-scale end-user applications and services are developed, tested, deployed and operated. A rich ecosystem of Cloud computing services and providers has appeared. This complex environment provides choices, but also prose challenges to engineers to mash-ups and comparative. Some of the key characteristics include [11]:

- Network access and management infrastructure
- Availability and ubiquitous web access
- Centralized feature updating, which avoids the need for downloadable patches and upgrades
- Centralized application delivery model, including architecture, pricing, partnering, and management characteristics;

An integrated Cloud computing reference model has been proposed by Lenk et al. [12] as Figure 1. It can categorize the Cloud computing services on the basis of distinct service features. On the lowest level of the Cloud infrastructure is the infrastructure as a service (IaaS). Providing computing and storage infrastructure in a centralized, location-transparent service. The platform as a service (PaaS) level included the Programming Environment and Execution Environment, and the software as a service (SaaS) layer locates the applications that run on the Cloud and provide services to customers.

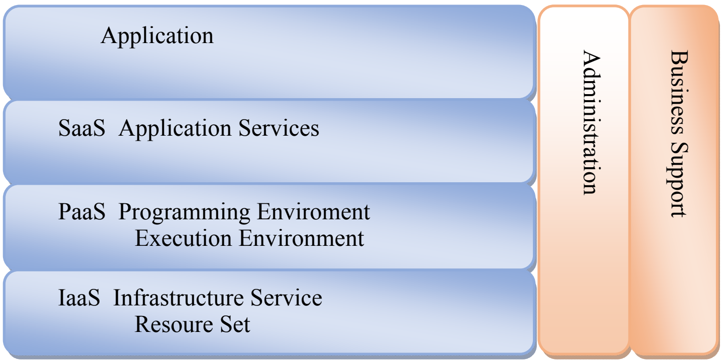
Application
SaaS Application Services
PaaS Programming Enviroment
Execution Environment
IaaS Infrastructure Service
Resoure Set
Administration
Business Support

Fig. 1. The Layered Services [12]

However, there are no security standards specific to Cloud computing [13], and security often addressed as an afterthought in the rush to adopt those technologies. The data in a cloud ranges from public source, which has minimal security concerns. For highly sensitive information, the public trust that surrounds the handling of this data reflects an assumed level of high security. If a business's primary function is to provide services that require sensitive data, then the security that they expect or need is higher than that required for a business that processes none-sensitive information. In our opinion, security services should be included in the IaaS layer to support the whole Cloud computing environment. There are seven security issues - privileged user access, regulatory compliance, data location, data segregation, recovery, investigative support, and long-term viability [14] should be developed. As we know, the first one of the above issues is about "authentication". "You need to be able to log into the service, you need to give people a route to access it, you need protocols to send traffic to it" said Ken Munro, managing director of SecureTest [15]. It is probable for attackers to intercept the process of data transmission or access to the user interface. To tackle the illegal access and to prevent the information from being stolen during transmission, a password-based user authentication scheme is proposed for information integrity. Such authentication problem should be solved in the first priority. Recently, some scholars proposed authentication schemes [16,17]. And some propose password authentication scheme for the Telecare medicine information system and demonstrated that it can withstand various attacks [18,19]. However, those proposed schemes only suitable for single environment. Each one needs to construct the architecture for authentication. In this paper, we propose an authentication-as-a-services (AaaS) architecture to provide trust and anonymity for Cloud computing.

2 The proposed Authentication-as-a-Services (AaaS) Scheme

2.1 Overview

We propose a privacy-preservation authentication scheme for Cloud computing architecture (Figure 2). In this scheme, users log into an authentication-as-a-service (AaaS) provider to get a token. With this token, user can roam in various everything-as-a-service (XaaS) providers without re-registration and re-authentication. It means that users do not have to leave sensitive information behind in different XaaS. The risk can be reduced for the Cloud computing service providers.

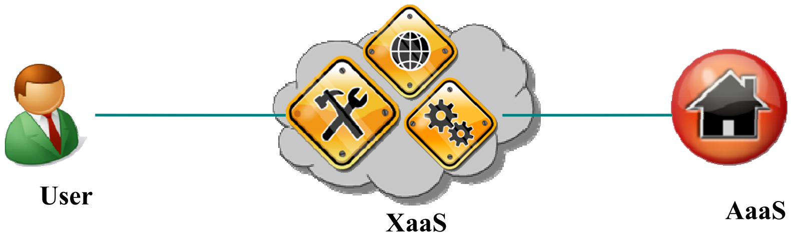

Fig. 2. System overview

All of parameters mentioned in this scheme are defined as following notations in Table1.

Table 1. Notations

Notation	Meaning
ID_i	The identification of the user *i*
PW_i	The Password of the user *i*
SID_j	The identity of XaaS *j*
TKN$_i$	The token for authorized user *i*
x	The permanent token key of *AaaS*
K_{AX}	The pre-shared key of AaaS and XaaS
n_i	A random number
$E_x(\,)$	A symmetric-key algorithm using key x to encrypt a message
$D_x(\,)$	A symmetric-key algorithm using key x to decrypt a message
$Ep_i(\,)$	A asymmetric-key algorithm using *i*'s public key to encrypt a message
$Dp_i(\,)$	A asymmetric-key algorithm using *i*'s public key to decrypt a message
$h(\cdot)$	Hash function
$\oplus$	Exclusive-or operation

2.2 Authentication Phase

As a user login to AaaS for authentication, the authorized user will get a token for login.

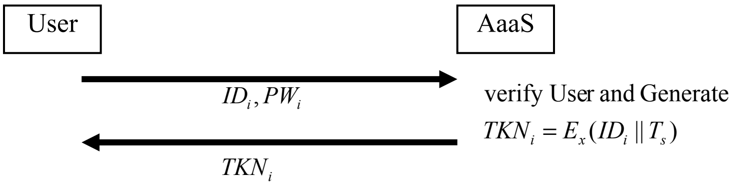

Fig. 3. Authentication Phase

Step 1. The user login to AaaS with his account ID_i and password PW_i .

Step 2. Suppose the user's account has been verified, the following encrypted token is generated by the user's identity ID_i and the token's due time T_s.

$$TKN_i = E_x(ID_i \,\|\, T_s) \tag{1}$$

The token is sent back to the user.

2.3 Service Commitment Phase

As XaaS is asked for services by a user, the user's token is used for anonymous authentication (Figure 4). Mutual authentication is achieved among AaaS, XaaS, and the user in this phase.

Step 1. The user's token TKN_i , XaaS's identity SID_j and the request time t_i are concatenated and encrypted by AaaS's public key PK_{AaaS} for generating the following request message.

$$req_{Ui} = Ep_{PK_{AaaS}}(TKN_i \,\|\, SID_j \,\|\, t_i) \tag{2}$$

The message req_{Ui} is sent to XaaS.

Step 2. XaaS generates another request message req_{Xj} by the key K_{AX} and a nonce n_j.

$$req_{Xj} = K_{AX} \oplus n_j \tag{3}$$

Then XaaS sends (req_{Ui}, SID_j, req_{Xj}) to AaaS.

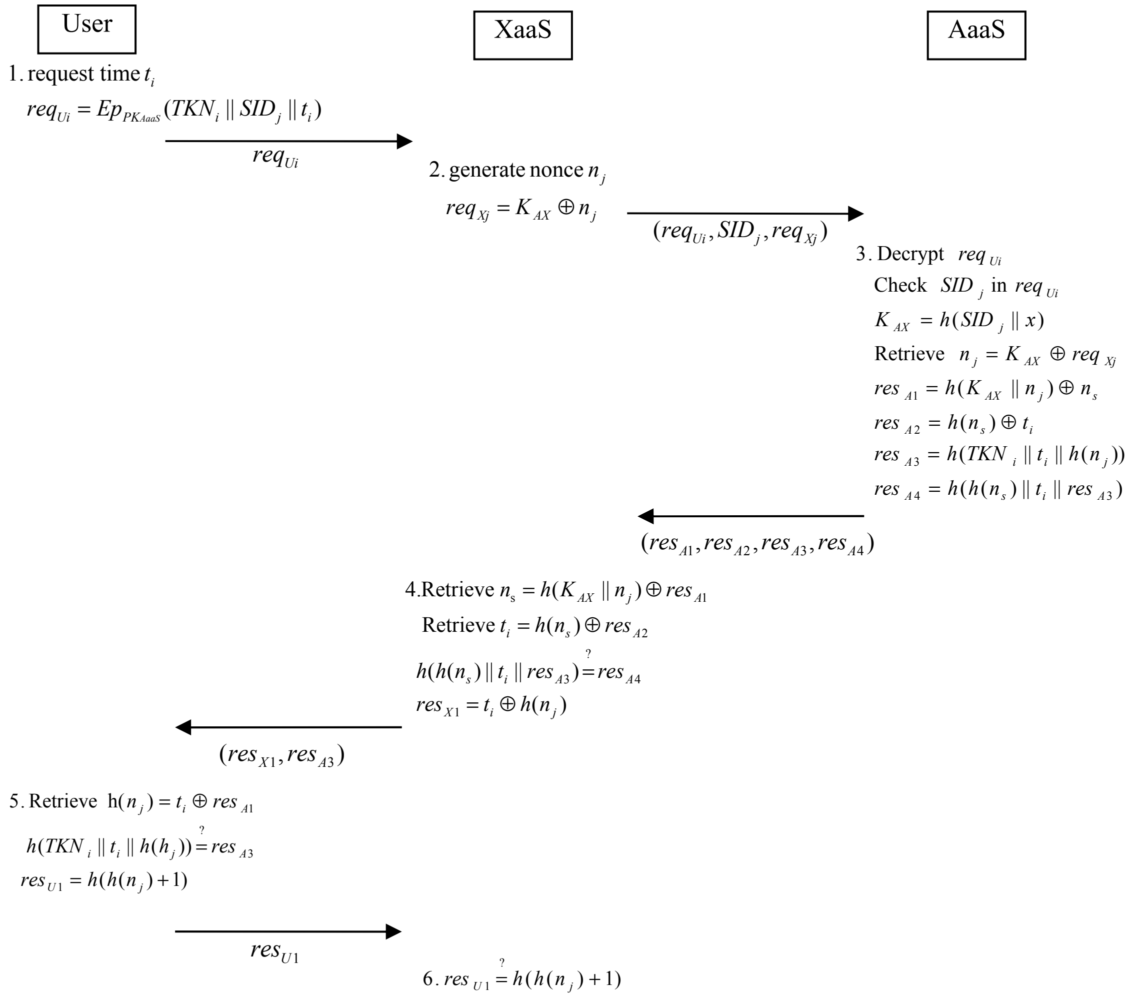
User
XaaS
AaaS
1. request time t_i
$req_{Ui} = Ep_{PK_{AaaS}}(TKN_i \| SID_j \| t_i)$
req_{Ui}
2. generate nonce n_j
$req_{Xj} = K_{AX} \oplus n_j$
$(req_{Ui}, SID_j, req_{Xj})$
3. Decrypt req_{Ui}
Check SID_j in req_{Ui}
$K_{AX} = h(SID_j \| x)$
Retrieve $n_j = K_{AX} \oplus req_{Xj}$
$res_{A1} = h(K_{AX} \| n_j) \oplus n_s$
$res_{A2} = h(n_s) \oplus t_i$
$res_{A3} = h(TKN_i \| t_i \| h(n_j))$
$res_{A4} = h(h(n_s) \| t_i \| res_{A3})$
$(res_{A1}, res_{A2}, res_{A3}, res_{A4})$
4.Retrieve $n_s = h(K_{AX} \| n_j) \oplus res_{A1}$
Retrieve $t_i = h(n_s) \oplus res_{A2}$
$h(h(n_s) \| t_i \| res_{A3}) \overset{?}{=} res_{A4}$
$res_{X1} = t_i \oplus h(n_j)$
(res_{X1}, res_{A3})
5. Retrieve $\mathrm{h}(n_j) = t_i \oplus res_{A1}$
$h(TKN_i \| t_i \| h(h_j)) \overset{?}{=} res_{A3}$
$res_{U1} = h(h(n_j)+1)$
res_{U1}
6. $res_{U1} \overset{?}{=} h(h(n_j)+1)$

Fig. 4 Service Commitment Phase

Step 3. As AaaS received above message, it decrypts req_{Ui} by its secret key, $Dp_{SK_{AaaS}}(req_{Ui})$, to get the user's token TKN_i, XaaS's identity SID_j and the request time t_i. XaaS's identity SID_j should be confirmed since it is announced by the user and XaaS respectively. The request time t_i should be also confirmed as it is in the reasonable time limit.

Therefore, the token TKN_i will be decrypted by token key, $(ID_i \| T_s) = D_x(TKN_i)$, to get the user's identity ID_i and the token's due time T_s. And AaaS retrieve n_j from req_{Xj}. Only the T_s is not over due, AaaS generates another nonce n_s, and following messages.

$$\begin{aligned} res_{A1} &= h(K_{AX} \| n_j) \oplus n_s \\ res_{A2} &= h(n_s) \oplus t_i \\ res_{A3} &= h(TKN_i \| t_i \| h(n_j)) \\ res_{A4} &= h(h(n_s) \| t_i \| res_{A3}) \end{aligned} \qquad \textbf{(4)}$$

Then the message (res_{A1}, res_{A2}, res_{A3}, res_{A4}) is sent to XaaS.

Step 4. As XaaS receives above message, XaaS can get the nonce n_s and the request time t_i as follows.

$$\begin{aligned} n_s &= h(K_{AX} \| n_j) \oplus res_{A_1} \\ t_i &= h(n_s) \oplus res_{A_2} \end{aligned} \qquad \textbf{(5)}$$

And these two values' integrity can be verified as follows.

$$h(h(n_s) \| t_i \| res_{A3}) \overset{?}{=} res_{A4} \quad \textbf{(6)}$$

If the above equation hold, it means that XaaS has authenticate AaaS, and the correctness of the pre-share Key K_{AX} has confirmed.

Finally, XaaS generate following response message res_{X1} .

$$res_{X1} = t_i \oplus h(n_j) \quad \textbf{(7)}$$

Then the message (res_{X1}, res_{A3}) is sent to the user.

Step 5. As above message is received, the user retrieves h (n_j) from res_{X1} as follows.

$$h(n_j) = res_{X1} \oplus t_i \quad \textbf{(8)}$$

And the response message can be verified as follows.

$$h(TKN_i \| t_i \| h(n_j)) \overset{?}{=} res_{A3} \quad \textbf{(9)}$$

If the above equation hold, the user can confirm that the response message is generated by trust AaaS and authorize XaaS. Then user generates the response message res_{U1} to XaaS as follows.

$$res_{U1} = h(h(n_j) + 1) \quad \textbf{(10)}$$

Step 6. As above message is received, XaaS verify the message res_{U1} as follows.

$$res_{U1} \overset{?}{=} h(h(n_j) + 1) \quad \textbf{(11)}$$

Only now is the legitimacy of the user manifested, XaaS ready for executing the user's Cloud computing request.

3 Analysis

The analysis is divided into several parts as the condition mentioning before. More detail of the analysis and how our system can conquer the threat will be described as follows.

3.1 A novel business model

A novel business model on cloud computing service is proposed in this paper. Users are kept anonymous and untraceable on using the services on XaaS. Users do not have to leave sensitive information behind in XaaS but still can roaming in various cloud computing services. Courses will not store or get any user's sensitive data. The "authentication" can be provided by AaaS as a kind of cloud computing services. AaaS can provide an effective authentication infrastructure in a centralized, location-transparent service. It will be attractive, interesting, and convenient for users and engineers in cloud computing services.

3.2 Anonymity

As the user requests for the services of XaaS, the request message req_{Ui} is encrypted with AaaS's public key.

$$req_{Ui} = Ep_{PK_{AaaS}}(TKN_i \| SID_j \| t_i) \quad \textbf{(12)}$$

It is impossible for anyone to get the user's token. Only AaaS can use its secret key SK_{AaaS} to decrypt req_{Ui} to get user's token TKN_i , XaaS's identity SID_j, and time t_i.

$$(TKN_i \| SID_j \| t_i) = Dp_{SK_{AaaS}}(req_{Ui})$$

The user's identity or even the user's token will not be revealed in our scheme. Such design can avoid XaaS to make any connection between users' data and his identity.

3.3 Mutual Authentication

Mutual authentication can be achieved amount AaaS, XaaS, and the user in our design. All transmitted message are well-protected. Without knowing the secret information, the attacker cannot obtain any useful information from the transmitted message. Since only AaaS can decrypt the user's request *req_{Ui}* by its secret key $SK_{AaaS,}$ and knows the corresponding secret information, ie user's token TKN_i and pre-share key. The response message (res_{A1}, res_{A2}, res_{A3}, res_{A4}) can ensure the legitimacy of AaaS. Similarly, user can ensure the legitimacy of AaaS once the following equation verified successfully.

$$h(TKN_i \| t_i \| h(n_j)) \overset{?}{=} res_{A3}$$

After, only authorized XaaS and users can actually proceed with the following authentication process. Moreover, with the response message res_{X1} and res_{U1}, both of XaaS and the user can authenticate with each other. Hence the mutual authentication will be achieved in our design.

3.4 Simplicity and Practicability

In this architecture, XaaS do not need to maintain authentication procedure, and can depend on AaaS to authenticate the user. It can reduce the cost to maintain a secure environment for users' sensitive information. This design is easy to be applied to the current cloud computing system without need of extra infrastructures. For a cloud computing users, they can roaming in various cloud computing services without leaving sensitive information behind. Users are kept anonymous and untraceable on using the services on XaaS. It will attract users to use the XaaS without the need to convert the trust and privacy problem. Therefore, our design can bring more users to XaaS.

4 Conclusions

In this paper, we propose a design of authentication-as-a-services (AaaS) architecture in cloud computing. We analyzed the propose business model, anonymity, mutual authentication, and simplicity and practicability. In our design, user's sensitive information will not leak to XaaS. Moreover, user can roaming in various XaaS providers without re-registration and re-authentication. It will be attractive, interesting, and convenient for users and engineers in cloud computing services.

References

[1] D. Baran, "Cloud Computing Basics," http://www.webguild.org/20080729/cloud-computing-basics, seen: 2012-09-15.

[2] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," *Future Generation Computer Systems*, Vol. 25 No. 6, pp. 599-616, 2009.

[3] Google. http://www.google.com

[4] Facebook. http://www.facebook.com

[5] RealityServer. Nvidia. http://www.nvidia.com/object/realityserver.html, seen: 2011-06-15.

[6] Amazon Elastic Compute Cloud (EC2). http://www.amazon.com/ec2/

[7] Amazon Simple Storage Service (S3). http://www.amazon.com/s3/

[8] Google App Engine. http://appengine.google.com

[9] Microsoft Azure. http://www.microsoft.com/azure/

[10] Oracle Sun Cloud Computing. http://www.oracle.com/us/solutions/cloud/overview/index.html

[11] E. Traudt, and A. Konary, *Software-as-a-Service Taxonomy and Research Guide*, Technical report 601.06, 2005.

[12] M. Lenk, J. Klems, S. Nimis, S. Tai, T. Sandholm, “What's Inside the Cloud? An Architectural Map of the Cloud Landscape,” in *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, ACM Press, pp. 23-31, 2009

[13] S. Mansfield-Devin, “Danger in the Clouds,” *Network Security,* Vol. 2008, No. 12, pp. 9-11, 2008.

[14] J. Heiser and M. Nicolett, *Assessing the Security Risks of Cloud Computing*, Gartner Report, 2008.

[15] Security Testing, Audit & Compliance - NCC Group, http://www.securetest.com/

[16] T.Y. Wu and Y.M. Tseng, “An Efficient User Authentication and Key Exchange Protocol for Mobile Client-Server Environments,” *Computer Networks*, Vol. 54, No. 9, pp. 1520-1530, 2010.

[17] C. Guo, C.C. Chang, C.Y. Sun, “Chaotic Maps-Based Mutual Authentication and Key Agreement using Smart Cards for Wireless Communications,” *Journal of Information Hiding and Multimedia Signal Processing*, Vol. 4, No. 2, pp. 99-109, 2013.

[18] Z. Y. Wu, Y. C. Lee, F. Lai, H. C. Lee, Y. F. Chung, “A Secure Authentication Scheme for Telecare Medicine Information Systems,” *Journal of Medical Systems*, Vol. 36, No. 3, pp.1529-1535, 2012.

[19] D. He, J. Chen, R. Zhang, “A More Secure Authentication Scheme for Telecare Medicine Information Systems,” *Journal of Medical Systems*, Vol. 36, No. 3, pp. 1989-1995, 2012.