Distributed Design of a Central Service to Ensure Deterministic Behavior

IMRAN ALI JOKHIO*, SANA HOOR JOKHIO**, AND JAVED ALI BALOCH***

RECEIVED ON 18.11.2011 ACCEPTED ON 21.06.2012

ABSTRACT

A central authentication service to EPC (Electronic Product Code) system architecture is proposed in our previous work. A challenge for a central service always arises that how it can ensure a certain level of delay while processing emergent data. The increasing data in the EPC system architecture is tags data. Therefore, authenticating increasing number of tag in the central authentication service with a deterministic time response is investigated and a distributed authentication service is designed in a layered approach. A distributed design of tag searching services in SOA (Service Oriented Architecture) style is also presented. Using the SOA architectural style a self-adaptive authentication service over Cloud is also proposed for the central authentication service, that may also be extended for other applications.

Key Words: **Quality of Service, Trusted Naming Service, User Registration Repository, Central EPCIS Repository, Supply Chain Management, Reader Interface, and Security Method Operations Store.**

1. INTRODUCTION

RFID (Radio Frequency Identification) Technology has a number of applications. Security and privacy challenges [1-2] to this technology are a major hindrance in successful implementation. A number of security methods have been proposed globally to protect the confidential data on RFID tags [3]. But, exhaustive authentication searching in the back-end servers poses a new challenge. This exhaustive authentication-searching problem in EPC system architecture [4-5] is introduced and explained in [6-8]. A central trusted naming service is also proposed in [8]. In this paper, our previous work is extended to design a distributed authentication service. In order to design a distributed authentication service, layered architecture approach is adopted to separate number of processes of authentication service. This ensures the overall QoS (Quality of Service) (deterministic response time) of the authentication service workflow. The number of tags in a global RFID system is emergent, hence SOA style designed is presented to achieve a self-adaptive service whilst exploiting the on-demand service availability of the Cloud computing infrastructure.

2. SOA

In todays' business applications, the SOA architectural style [9] is becoming more popular than ever. The popularity of the SOA architectural style is owed to its loosely coupled system components and their interaction capabilities. As a unit, these components may offer services and dynamic selection of a service for a workflow to ensure

* Assistant Professor, Department of Software Engineering, Mehran University of Engineering & Technology Jamshoro.
** Assistant Professor, Department of Computer Systems Engineering, Mehran University of Engineering & Technology Jamshoro.

a level of QoS. An application or business process may have certain challenges, especially from a service providers' view i.e. if we as a provider are offering a service that is a part of a workflow of our clients or our own application, it may have to ensure a SLA (Service Level Agreement) in which response time is the main constraint of the QoS. For this scenario, in Fig. 1 elements of SOA are shown as business processes, applications and services. Business process may have certain requirements and one of these requirements is response time. The time critical business processes result in deterministic applications. For deterministic applications choreography of services is very important [10]. Though traditionally, response time of a service depends on the computation of an operation in a service, for emergent data applications, response time of an operation of a service becomes a function of computation over data sources/sets. An example of this type of application is exhaustive searching or match and find computation. In run time or dynamic selection, to meet a SLA over emergent data sources/sets is a challenge for a service. The services are normally part of SOA application workflows, and any failure to meet a SLA will come forward as a catastrophe for a time critical application.

2.1 Self-Adaptive Services in SOA Architectural Style

A service is the basic entity responsible for maintaining the response time of a workflow. As discussed earlier, a

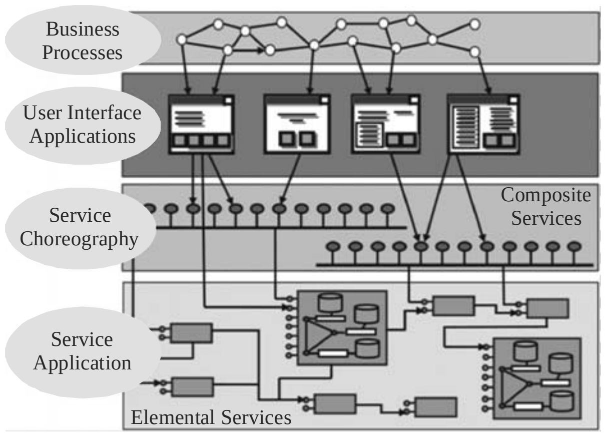

FIG. 1. SERVICE ORIENTED ARCHITECTURE PRACTICAL PORTAL

service having an exhaustive process over data would decrease its QoS with increasing data sets. In order to overcome this challenge, a design of a self adaptive service is described which can be configured to offer a certain level of QoS i.e. response time requirement with increasing data sets.The design of the central authentication service has certain access points to integrate it with existing EPC system architecture [8]. These access points have anumber of processes and the processes can be accessed by their respective interfaces as discussed in [8]. A layered approach to group the processes and the repositories while these processes are being accessed is shown in Fig. 2. The ABS (Authentication Broker Service) architecture has three layers: definition, management and authentication. From a logical view of the TNS (Trusted Naming Service) architecture, for all E_i the tags $T_{E_{ir}}$ are kept in CTR (Central Tag Repository) and the processing during authentication is the same for a tag $T_{E_{ir}}$ i.e. $P = \Sigma_{i=1}^{n} \Sigma_{r=1}^{k} p\left(T_{E_{ir}}\right)$ but in TNS rather than distributed across E_i. This does not allow the leakage of any information at all during any of process in the EPC networksystem architecture. It may be seen that with the ABS based EPC network architecture, the privacy and confidentiality of the RFID data is maintained.

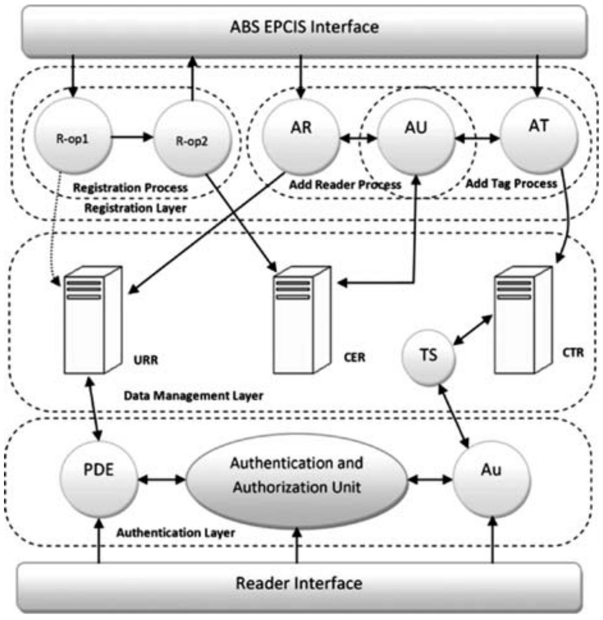

FIG. 2. ABS LAYERED ARCHITECTURE

But scalability was another problem in MORIS (Multi Organizational RFID System), so in the data management layer it is separated from the rest of the processes. This facilitates a flexible way to address this issue. For an authentication service, the response time or deterministic nature is the parameter of scalability so the data management layer can be enhanced to ensure the response time of the TS (Tag Search) during the authentication process. The scalability issue is mainly due to TS process but in the proposed design TS is separated from other operations. In the data management layer, the TS process is carried on the CTR repository and the authentication layer accesses the URR (User Registration Repository) and the CER repositories to enforce policies. So a distributed design of CTR and TS process can provide improved scalability. The design of the ABS is a loosely coupled architecture where the processes and operations are operating independently. This ensures the deterministic behavior of authentication in the TNS.

In Fig. 3, SS (Self-Adaptive Service) components are shown. In order to maintain a deterministic behavior of services, a SOA architectural style [9-10] based model is designed to deploy services. In this model for deterministic services, there is a service interface for clients, which is very same as of any service. The design model has a core component, SSM (Self-Adaptive Service Manager), which gets a feedback (response delay) from the SS. A response time delay requirement is also defined in SSM by a SLA so when a SS time response delay reaches the defined threshold, the SSM triggers a call to the SR (Service Replicator). SR replicates the service and deploys a newly created replicated service and an interface to it is sent to SSM. In this way the SSM has all the interfaces of the internal service in the SS design. When a client queries the service interface, it forwards the request to the SS services and all the deployed services process chunks of the data sets. The response time delay will be within the threshold defined as the SSM services are managed to do so. This self-adaptive approach to creating and deploying the SS continues to scale up and provides the required QoS. The self-adaptive scaling up of the SS is needed as new data sets may be added to data sources. An example of this type of application is an RFID based SCM system. The addition of tags and readers is very common in RFID based SCM (Supply Chain Management) and will increase the number of tags in the RFID data events repository. This will increase authentication delay because of tag searching delays. But a dynamic, self-adapting scaling up of a service may maintain a certain level QoS of response time delays.

3. EXHAUSTIVE SEARCHING

The authentication process begins when a reader queries the broker about a tag by accessing the RI interface as shown in Fig. 4 and explained in our previous work Lemma-2 [8]. The SMOS component of the ABS searches a tag by encrypting and matching the tags. The encryption of the tag ID or EPC is as: f=h(x,ID,y) where *x* and *y* are two random numbers generated by reader and tag respectively. The encryption model h is usually one-way hash function or some other method having similar properties. SMOS receives *x*,*y* and f so to get the ID of a tag there is no searching technique but a blind search i.e. encrypting the

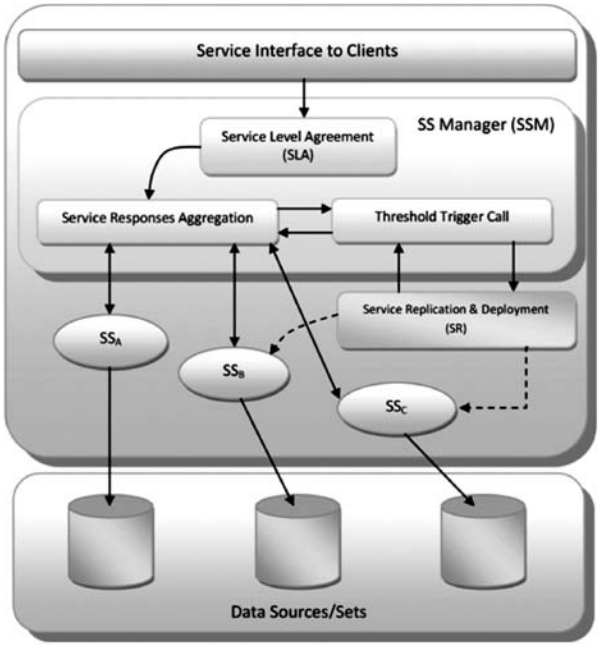
Service Interface to Clients
SS Manager (SSM)
Service Level Agreement (SLA)
Service Responses Aggregation
Threshold Trigger Call
Service Replication & Deployment (SR)
SS_A
SS_B
SS_C
Data Sources/Sets

FIG. 3. SELF-ADAPTIVE SERVICE DESIGN IN SOA ARCHITECTURAL STYLE

tags registered in CTR {z_1, z_2, z_3..z_n} as: $g=h(x,z_i,y)$ and matching $f=g$ until a match is found. This results in a brute force searching in the authentication process.

3.1 ABS Over Cloud

In the ABS, the SMOS repository is a separate repository to the policy definitions. This is helpful in separating the searching overhead of the encrypted EPC data caused by any security method deployed between a tag and a reader. The searching overhead of a tag may be maintained for a deterministic time response by exploiting the capabilities of the cloud-computing paradigm. We propose a self-adaptive distributed design approach for tag searching in CTR to maintain a certain level of QoS as shown in Fig. 4. It may be noted here that we assume all the components and interactions between these components in the Cloud is secure.

In order to maintain deterministic behavior of the authentication service a model is proposed to deploy the authentication service of the EPC network over the cloud. The cloud based authentication service has an ABS manager. It has a core component SM (Service Manager), which gets a feedback (authentication delay) from SMOS of the ABS. An authentication delay requirement is also defined in the SM so when the SMOS authentication delay reaches the defined threshold, the SM triggers a call to the Cloud to create a new VM (Virtual Machine). An ABS is deployed on the newly created VM and an inter-face to it is sent to the RI. In this way the RI has all the interfaces of the VM having ABS in the Cloud. When a reader queries the RI for the authentication of a tag, it forwards the request to all the VMs and ABS service hosted on a VM that holds the tag replies with a positive authentication reply. The authentication delay will be within the thresh- old defined as the VMs are managed to do so. However, the rest of the VMs will have redundant authentication processing, as these do not contain the tag. This redundancy cannot be avoided because of the nature of the security method on the tag as described in previous work [7-8]. This self-adaptive approach to create and deploy the ABS continues to scale up and provides the required QoS of the authentication. The self-adaptive scaling up of the ABS is needed as a new EPCIS service may register with the ABS and adds up its tags and readers or existing registered EPCIS services may add up their tags and readers. The addition of tags and readers is very common in RFID based systems and it will increase the number of tags in the CTR. This will increase authentication delay because of the SMOS tag searching delays. But with dynamic self-adaptive scaling up of the ABS over the Cloud may maintain a certain level QoS authentication delay.

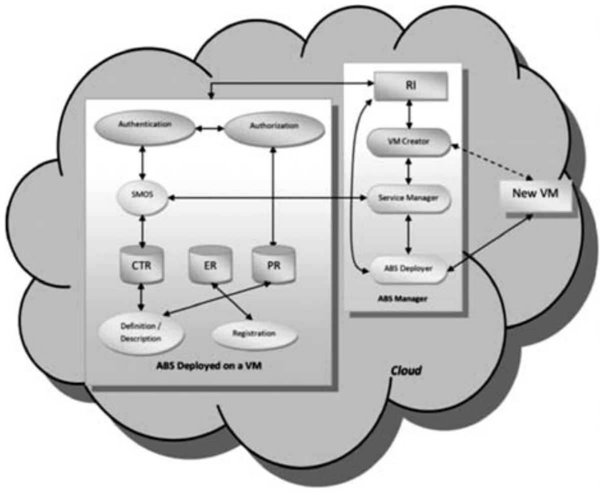

FIG. 4. SELF-ADAPTIVE DESIGN OF ABS OVER CLOUDS

4. EXPERIMENTAL EVALUATION

To demonstrate scalability of the proposed distributed design of TNS over large data with increased queries a number experiments are conducted.

4.1 Reliability Analysis

For reliability analysis experiments are performed multiple time. In section 5, the experiments are executed for three times, the standard deviation error is calculated and 90% confidence interval is calculated. In the scalability experiments, in order to calculate standard deviation error and show 90% confidence level, the experiments of each scenario is performed four times.

4.2 Experimental Setup/Configurations

In the second part of the experiments, the Cloud is set up using the Xen-Hypervisor 3.4. The hypervisor is configured with Ubuntu Kernel Version 2.6.31.4 having memory of 128 megabyte. Two virtual machines were configured in this setup each having the 1 Giga-byte of the RAM (Random Access Memory). Each virtual machine was running Ubuntu 8.0 operating system with Apache Axis2 platform and postgreSQL database configured on them.

4.3 Hardware

In the case of without Cloud setup, the experiments were performed on a single machine having features a dual-core processor and 2 Gigabytes of RAM, with hosting all the services and repositories of the EPC network architecture. While in the case of the Cloud setup a machine having features a dual-Xeon core processor and 4 gigabyte of RAM is used in the experiments.

5. SCALABILITY EXPERIMENTS

The scalability of the proposed TNS based EPC network is measured in terms of response time of the authentication service. In order to assure a response time, there is a need to assess the average authentication response time of an authentication service, so that each authentication request can be responded within a specified time. For this, the simulation of the TNS based EPC network is experimented with two scenarios whilst increasing the number of tags in the system, to demonstrate how the average authentication delay and response time of TNS behave. Experiments are run four time for each scenario. In order to perform reliability analysis of the experiment standard deviation error and 90% confidence interval is also calculated.

5.1 First Scenario

In the first scenario of TNS, all the tags of the system are kept in a single central repository and an authentication service is implemented over it. The experiment is done with varying number of tags as shown in Table 1. The experiments were repeated for several times for the reliability analysis. Table 1 shows the four runs and their respective authentication. The average authentication delay, standard deviation error and 90% confidence level interval is shown Table 2.

In Fig. 5 the four run's and the average authentication delay whilst increasing the total number of tags in the TNS are plotted. It can be noted in the Fig. 5, that the average authentication delay curve is of parabolic fashion rather than being linear. This indicates that, a level of average response time t can be maintained in the TNS for n tags. However, in a real-time application, the response time of a service needs to be assured. The next experiment is done by considering the response time requirement of a real-time application is t_{req} and $t>t_{req}$, in this case, the repository of the TNS can be divided into smaller chunks so that the worst time response twor should be less than t_{req}. And, an authentication service is replicated on all the chunks of the TNS repository. Only one replicated service across the chunks of the repository, will ultimately respond

TABLE 1. AUTHENTICATION DELAY IN TNS CENTRAL REPOSITORY

No. of Tags	Run-1 Delay (millisecond)	Run-2 Delay (millisecond)	Run-3 Delay (millisecond)	Run-4 Delay (millisecond)
1K	375.13283	462.0530049	400.5911598	436.5946751
5K	1262.47419	1342.541691	1285.925418	1319.090463
10K	2811.85355	2891.628441	2835.219074	2868.262916
15K	5186.12398	5327.013708	5227.389626	5285.748062
20K	7211.40345	7414.120734	7270.777968	7354.746216

in time t_{chunk} i.e. $t_{chunk} < t_{req}$. This response of the authentication service, over the chunk of the repository, is forwarded to meet the response time of the application. In order to demonstrate that division of the repository of the TNS into chunks can assure a certain level of response time requirement, a simulation of the self-adaptive TNS service over the virtualized resources is conducted and is discussed in the second scenario.

5.2 Second Scenario

In this scenario, in order to demonstrate that distributed design of the TNS using load balancing techniques can maintain a level of response time, the experiment of the first scenario is repeated by dividing the TNS central repository into two chunks. In a real world RFID application, this central repository grows with the increasing number of tags and the participating organizations. This increase in the number of tags in TNS needs to distribute the central repository internally. Not only that but there is also a need of the authentication and authorization service over this tag repository. The distribution of the central repository is to maintain the response time of the TNS service. Therefore in order to accommodate the distribution of the central repository, the virtual machines of the Clouds is a candidate solution for this problem. When a new virtual machine with a chuck of the central repository is created, the authentication and authorization service can easily be deployed over this virtual machine. However, a separate physical server may also be setup for the distribution of the central repository of the TNS service. But the virtual machines are relatively easier and faster to setup. The virtual machines also give an advantage of easier scaling up and shrinking down of the servers. Moreover, the resources of the virtual machines may not always be acquired by the TNS service; these resources may be used for other applications. Therefore, experiments of the distributed design of the TNS service are done using the Clouds. In this experiment two authentication services are replicated over the two virtual machines, each having a part of the central repository of the TNS. The responses of each of the deployed services are aggregated at a point that is accessible to the reader device.

Although in the design and implementation of the distributed SMOS, the central repository is divided into two equal parts, but because of the aggregation of the results in authentication service and virtual machines the authentication delay is not decreased to the half. Comparing the Table 1 and 3 it can be noted that the authentication delay may be improved with the proposed design of the distributed SMOS over virtual machines/ Clouds.

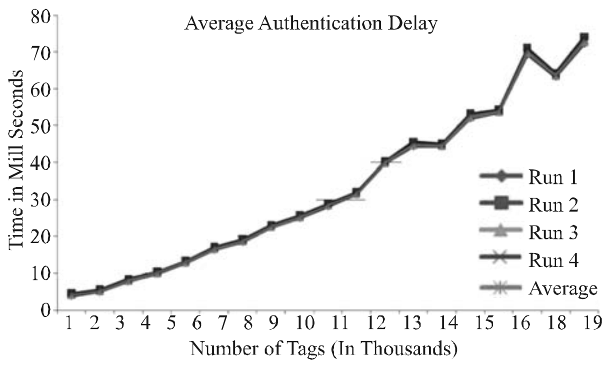

FIG. 5 .AVERAGE AUTHENTICATION DELAY

TABLE 2. STANDARD DEVIATION ERROR AND CONFIDENCE INTERVAL.

No. of Tags	Average Authentication Delay	Standard Deviation Error	Upper Interval	Lower Interval
1K	418.5929174	38.4087011	457.2896838	379.896151
5K	1302.50794	35.3806087	1338.790755	1266.225126
10K	2851.741	35.2513088	2887.891213	2815.590778
15K	5256.568844	62.2570246	5320.413423	5192.724265
20K	7312.762092	89.5776794	7404.624002	7220.900182

The experiment for second scenario was also repeated four time, the iterations of the experiment are shown in Table 3. Table 4 shows standard deviation error and 90% confidence level intervals to demonstrate the reliability of the performed experiments.

In the experimental simulations, the tags searching and matching time in the SMOS component of the TNS is shown in Fig. 5-6. It can be noted in the Fig. 5-6, that the average authentication delay of the tags verses increasing number of tags in CTR is decreased with the distributed deployment of the SMOS or central authentication repository. This is so because the searching and matching of the tags is are carried out separately in two parts of the SMOS repository (i.e. distributed SMOS). The distributed approach improves the response time of the authentication process. This demonstrates that the distributed design of SMOS and CTR can improve response time of the TNS and scale it up for the increasing number of tags. The distributed deployment of the SMOS repository can be extended, whilst dividing the central repository in more than two chunks too. Hence, in this way a certain level of response time of the TNS can be achieved. The increased number of virtual machines provides a scalable distributed design of the TNS because; the authentication services deployed over the virtual machines are independent of each other, as each service has its own data. For real-time deployment of the TNS that ensures a certain level of response time, a self-adapting approach may be incorporated. In the self-adaptive approach, a new virtual machine can be initiated on request to deploy an authentication service over a chunk of the central authentication repository. The data transfer from the central SMOS repository to a virtual machine is not an issue, as the virtual machines are kept in the single domain that has high-speed network bandwidth or connections. Furthermore, a new virtual machine is initiated when the existing virtual machines reach their maximum number of tags

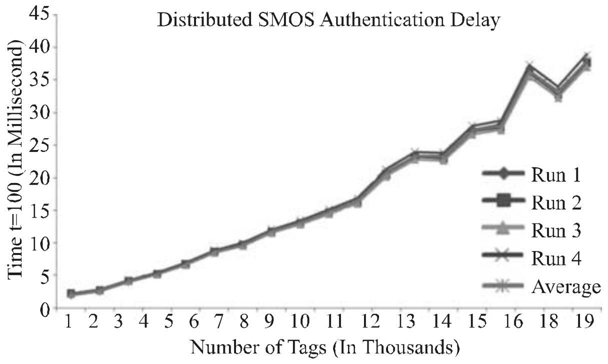

FIG. 6. DELAY IN DISTRIBUTED TNS OVER CLOUDS

TABLE 3. AUTHENTICATION DELAY IN TNS CENTRAL REPOSITORY

No. of Tags	Run-1 Delay (millisecond)	Run-2 Delay (millisecond)	Run-3 Delay (millisecond)	Run 4-Delay (millisecond)
1K	197.4383316	221.1908114	211.945781	217.8151724
5K	664.4601	673.9659253	659.4976913	690.1290904
10K	1479.922921	1468.42244	1443.919491	1517.189932
15K	2729.538937	2706.257265	2661.553845	2797.012106
20K	3795.4755	3765.358641	3702.66435	3890.673054

TABLE 4. STANDARD DEVIATION ERROR AND CONFIDENCE INTERVAL

No. of Tags	Average Authentication Delay	Standard Deviation Error	Upper Interval	Lower Interval
1K	212.0975241	38.4087011	250.7942905	173.4007577
5K	672.0132017	35.3806087	707.659165	636.3672384
10K	1477.363696	35.2513088	1512.87939	1441.848002
15K	2723.590538	62.2570246	2786.31449	2660.866586
20K	3788.542886	89.5776794	3878.792398	3698.29337

authentication data, so a new virtual machine's tags authentication data grows slowly. This does not create any sudden degradation of the authentication service.

6. CONCLUSION

In EPC system architecture TNS is a central service that has a deterministic response time as one of the prime requirement besides securing the confidential tags data. In this paper is aimed to the former requirement henceforth a distributed design of tag searching services in a SOA style is presented. Using the SOA architectural style a self-adaptive authentication service design is also proposed for the central authentication service. The proposed design of the service is aimed at the authentication service in EPC system architecture. But, the proposed work in this paper is a generic in its design that can be exploited in other application requiring a central scalable service. Therefore, the proposed design in this research work has wider candidate applications that may have deterministic response requirements. In our future work, the architecture presented in this work and Map-Reduce architecture are to be investigated for their applicability in Filtering and Blocking URLs system.

ACKNOWLEDGEMENT

Authors are thankful to Mehran University of Engineering & Technology, Jamshoro, Pakistan, to provide funding/ infrastructure to conduct this research in general and Department of Computer Systems & Software Engineering in particular.

REFERENCES

[1] Henrici, D., “RFID Security and Privacy: Concepts, Protocols, and Architectures”, Lecture Notes in Electrical Engineering, Springer Publishing Company, Incorporated, 2008.

[2] Neumann, P.G., "Risks to the Public", SIGSOFT Software Engineering Notes, Volume 32, No. 5, pp. 17-25, 2007.

[3] Langheinrich, M., "A Survey of RFID Privacy Approaches", Personal Ubiquitous Computing, Volume 13, No. 6, pp. 413-421, 2009.

[4] Kin Seong Leong, D.W.E., and MunLeng, N.G., "EPC Network Architecture", Auto-ID Center MIT, White Paper, 2005.

[5] Ken Traub, G.A., "EPC Network Architecture Framework", Auto-ID Center/EPCglobal, Technical Report. July, 2005.

[6] Jokhio, I.A., and Xu, J., "An Authentication Broker Service for Secure and Confidential EPC Code", 22nd International Symposium on Information, Communication and Automation Technologies, 2009.

[7] Jokhio, I.A., and Xu, J., "Data Privacy Management in a Multi-Organizational RFID Authentication", 6th International Conference on Wireless Communications, Networking and Mobile Computing, Chengdu, China, September, 2010.

[8] Jokhio, I.A., Jokhio, S.H., and Shaikh, F.K., "A Trusted Naming Service for Things of Internet" Mehran University Research Journal of Engineering & Technology, Volume 31, No. 2, pp. 325-334, Jamshoro, Paksitan, April, 2012.

[9] Lublinsky, B., "Defining SOA as an Architectural Style: Align Your Business Model with Technology", Technical Report, January, 2007. http://www.ibm.com/developerworks/architecture/library/ar-soastyle/ (Last Accessed February, 2012.

[10] Sprott, L.W.D., “Understanding SOA”, http://msdn.microsoft.com/en-us/library/aa480021.aspx, (Last Accessing February, 2012.