ISSN:2320-0790

An International Journal of Advanced Computer Technology

SCALABLE AND SECURE SHARING OF PERSONAL HEALTH RECORDS IN CLOUD COMPUTING USING ATTRIBUTE BASED ENCRYPTION

Sathini Santhosh Kumar[1], Shiva Prasad[2], M. Parimala[3], Prof.Dr.G.Manoj Someswar[4]

[1]M.Tech.(CSE), Tirumala Engineering College, Affiliated to JNTUH, Hyderabad, Telangana, India
[2]M.Tech.(CSE), Associate Professor, Tirumala Engineering College, Affiliated to JNTUH, Hyderabad, Telangana, India
[3]M.Tech.(CSE), Associate Professor, Tirumala Engineering College, Affiliated to JNTUH, Hyderabad, Telangana, India
[4]B.Tech., M.S.(USA), M.C.A., Ph.D., Director General & Scientist 'G', Global Research Academy, Hyderabad, Telangana, India

Abstract: Personal health record (PHR) is an emerging patient-centric model of health information exchange, which is often outsourced to be stored at a third party, such as cloud providers. However, there have been wide privacy concerns as personal health information could be exposed to those third party servers and to unauthorized parties. To assure the patients' control over access to their own PHRs, it is a promising method to encrypt the PHRs before outsourcing. Yet, issues such as risks of privacy exposure, scalability in key management, flexible access and efficient user revocation, have remained the most important challenges toward achieving fine-grained, cryptographically enforced data access control. In this research paper, we propose a novel patient-centric framework and a suite of mechanisms for data access control to PHRs stored in semi-trusted servers. To achieve fine-grained and scalable data access control for PHRs, we leverage attribute based encryption (ABE) techniques to encrypt each patient's PHR file. Different from previous works in secure data outsourcing, we focus on the multiple data owner scenario and divide the users in the PHR system into multiple security domains that greatly reduces the key management complexity for owners and users. A high degree of patient privacy is guaranteed simultaneously by exploiting multi-authority ABE. Our scheme also enables dynamic modification of access policies or file attributes, supports efficient on-demand user/attribute revocation and break-glass access under emergency scenarios. Extensive analytical and experimental results are presented which show the security, scalability and efficiency of our proposed scheme.

Keywords: Personal Health Record (PHR), Attribute Based Encryption (ABE), Fine-grained Data Access Control, Break- glass, PUD - public domains, PSD - personal domains, AA - attribute authority, MA-ABE - multi-authority ABE, KP-ABE - key policy ABE

I. INTRODUCTION

In recent years, personal health record (PHR) has emerged as a patient-centric model of health information exchange. A PHR service allows a patient to create, manage, and control her personal health data in one place through the web, which has made the storage, retrieval and sharing of the medical information more efficient. Especially, each patient is promised the full control of her medical records and can share her health data with a wide range of users, including healthcare providers, family members or friends. Due to the high cost of building and maintaining specialized data centers, many PHR services are outsourced to or provided by third-party service providers, for example, Microsoft HealthVault1. Recently, architectures of storing PHRs in cloud computing have been proposed in.

While it is exciting to have convenient PHR services for everyone, there are many security and privacy risk which could impede its wide adoption. The main concern is about whether the patients could actually control the sharing of their sensitive personal health information (PHI), especially when they are stored on a third-party server which people may not fully trust. On the one hand, although there exist healthcare regulations such as HIPAA which is recently amended to incorporate business associates, cloud providers are usually not covered entities. On the other hand, due to the high value of the sensitive personal health information (PHI), the third-party storage servers are often the targets of various malicious behaviors which may lead to exposure of the PHI. As a famous incident, a Department of Veterans Affairs database containing sensitive PHI of 26.5 million military veterans, including their social security numbers and health problems was stolen by an

employee who took the data home without authorization. To ensure patient-centric privacy control over their own PHRs, it is essential to have fine-grained data access control mechanisms that work with semi-trusted servers.

A feasible and promising approach would be to encrypt the data before outsourcing. Basically, the PHR owner herself should decide how to encrypt her files and to allow which set of users to obtain access to each file. A PHR file should only be available to the users who are given the corresponding decryption key, while remain confidential to the rest of users. Furthermore, the patient shall always retain the right to not only grant, but also revoke access privileges when they feel it is necessary However, the goal of patient-centric privacy is often in conflict with scalability in a PHR system. The authorized users may either need to access the PHR for personal use or professional purposes. Examples of the former are family member and friends, while the latter can be medical doctors, pharmacists, and researchers, etc. We refer to the two categories of users as personal and professional users, respectively. [1] The latter has potentially large scale; should each owner herself be directly responsible for managing all the professional users, she will easily be overwhelmed by the key management overhead. In addition, since those users' access requests are generally unpredictable, it is difficult for an owner to determine a list of them. On the other hand, different from the single data owner scenario considered in most of the existing works in a PHR system, there are multiple owners who may encrypt according to their own ways, possibly using different sets of cryptographic keys. Letting each user obtain keys from every owner who's PHR she wants to read would limit the accessibility since patients are not always online. An alternative is to employ a central authority (CA) to do the key management on behalf of all PHR owners, but this requires too much trust on a single authority (i.e., cause the key escrow problem). In this paper, we endeavor to study the patient centric, secure sharing of PHRs stored on semi-trusted servers, and focus on addressing the complicated an hallenging key management issues. In order to protect he personal health data stored on a semi-trusted server,

We adopt attribute-based encryption (ABE) as the main encryption primitive. Using ABE, access policies are expressed based on the attributes of users or data, which enables a patient to selectively share her PHR among a set of users by encrypting the file under a set of attributes, without the need to know a complete list of users. The complexities per encryption, key generation and decryption are only linear with the number of attributes involved. However, to integrate ABE into a large-scale PHR system, important issues such as key management scalability, dynamic policy updates, and efficient on-demand revocation are non-trivial to solve, and remain largely open up-to-date. To this end, we make the following main contributions:

(1) We propose a novel ABE-based framework for patient-centric secure sharing of PHRs in cloud computing environments, under the multi-owner settings. To address the key management challenges, we conceptually divide the users in the system into two types of domains, namely public and personal domains. In particular, the majority professional users are managed distributively by attribute authorities in the former, while each owner only needs to manage the keys of a small number of users in her personal domain. In this way, our framework can simultaneously handle different types of PHR sharing applications' requirements, while incurring minimal key management overhead for both owners and users in the system. In addition, the framework enforces write access control, handles dynamic policy updates, and provides break-glass access to PHRs under emergence scenarios.

(2) In the public domain, we use multi-authority ABE (MA-ABE) to improve the security and avoid key escrow problem. Each attribute authority (AA) in it governs a disjoint subset of user role attributes, while none of them alone is able to control the security of the whole system. We propose mechanisms for key distribution and encryption so that PHR owners can specify personalized fine-grained role-based access policies during file encryption. In the personal domain, owners directly assign access privileges for personal users and encrypt a PHR file under its data attributes. Furthermore, we enhance MA-ABE by putting forward an efficient and on-demand user/attribute revocation scheme, and prove its security under standard security assumptions. In this way, patients have full privacy control over their PHRs. [2]

(3) We provide a thorough analysis of the complexity and scalability of our proposed secure PHR sharing solution, in terms of multiple metrics in computation, communication, storage and key management. We also compare our scheme to several previous ones in complexity, scalability and security. Furthermore, we demonstrate the efficiency of our scheme by implementing it on a modern workstation and performing experiments/simulations.

Compared with the preliminary version of this paper there are several main additional contributions:

We clarify and extend our usage of MA-ABE in the public domain, and formally show how and which types of user-defined file access policies are realized.

We clarify the proposed revocable MA-ABE scheme, and provide a formal security proof for it.

We carry out both real-world experiments and simulations to evaluate the performance of the proposed solution in this research paper.

Existing System

In Existing system a PHR system model, there are multiple owners who may encrypt according to their own ways, possibly using different sets of cryptographic keys. Letting each user obtain keys from every owner who's PHR she wants to read would limit the accessibility since patients are not always online. An alternative is to employ a central authority (CA) to do the key management on behalf of all PHR owners, but this requires too much trust on a single authority (i.e., cause the key escrow problem).[3]

Proposed System

We endeavor to study the patient centric, secure sharing of PHRs stored on semi-trusted servers, and focus on addressing the complicated and challenging key

management issues. In order to protect the personal health data stored on a semi-trusted server, we adopt attribute-based encryption (ABE) as the main encryption primitive.
Using ABE, access policies are expressed based on the attributes of users or data, which enables a patient to selectively share her PHR among a set of users by encrypting the file under a set of attributes, without the need to know a complete list of users.
The complexities per encryption, key generation and decryption are only linear with the number of attributes involved.

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
2.Select methods for presenting information.
3.Create document, report, or other formats that contain information produced by the system.
The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

SYSTEM DESIGN

Figure 1: Data Flow Diagram

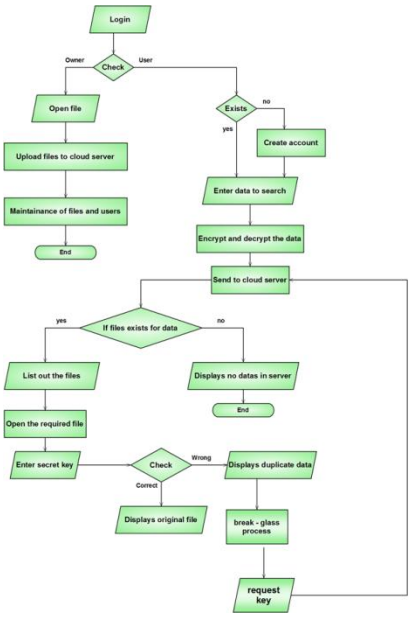


Figure 2: Use Case Diagram

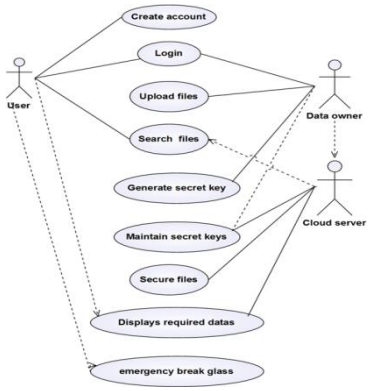


Figure 3: Class Diagram

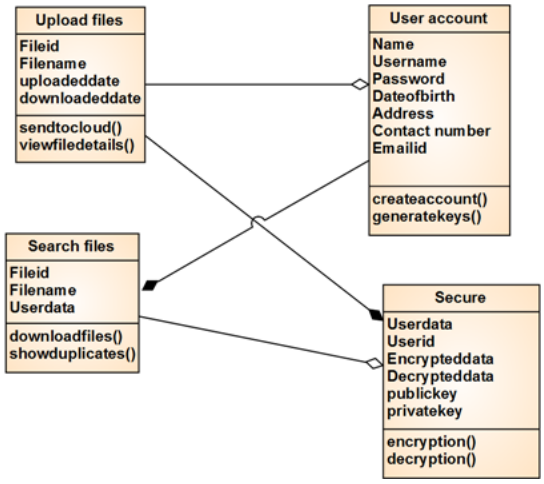


Figure 4: Sequence Diagram

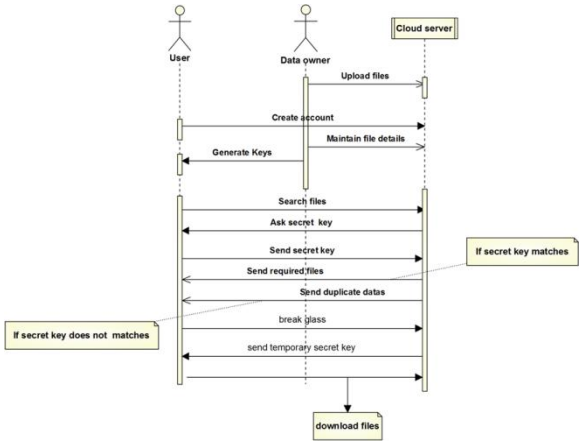
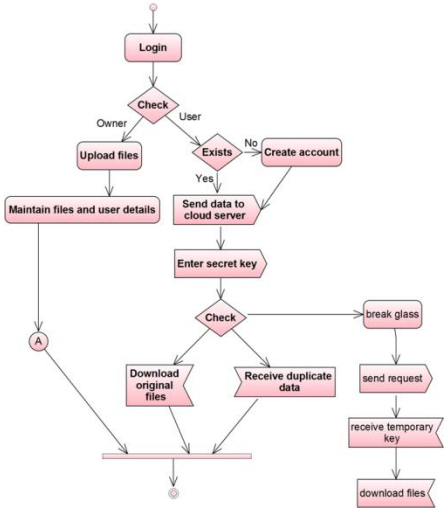


Figure 5: Activity Diagram



SYSTEM STUDY

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.[4]

Three key considerations involved in the feasibility analysis are

ECONOMICAL FEASIBILITY
TECHNICAL FEASIBILITY
SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased. [5]

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system. [6]

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.[7]

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. [8] It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.[9]

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input: identified classes of valid input must be accepted.

Invalid Input: identified classes of invalid input must be rejected.

Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.[10]

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.[11]

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.[12]

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.[13]

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.[14]

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

All field entries must work properly.

Pages must be activated from the identified link.

The entry screen, messages and responses must not be delayed.

Features to be tested:

Verify that the entries are of the correct format

No duplicate entries should be allowed

All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.[15]

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.[16]

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

Modules

Registration

Upload files

ABE for Fine-grained Data Access Control

Setup and Key Distribution

Break-glass

Modules Description

Registration

In this module normal registration for the multiple users. There are multiple owners, multiple AAs, and multiple users. The attribute hierarchy of files – leaf nodes is atomic file categories while internal nodes are compound categories. Dark boxes are the categories that a PSD's data reader has access to.

Two ABE systems are involved: for each PSD the revocable KP-ABE scheme is adopted for each PUD, our proposed revocable MA-ABE scheme.

PUD - public domains

PSD - personal domains

AA - attribute authority

MA-ABE - multi-authority ABE

KP-ABE - key policy ABE

Upload files

In this module, users upload their files with secure key probabilities. The owners upload ABE-encrypted PHR files to the server. Each owner's PHR file encrypted both under a certain fine grained model.

ABE for Fine-grained Data Access Control

In this module ABE to realize fine-grained access control for outsourced data especially, there has been an increasing interest in applying ABE to secure electronic healthcare records (EHRs). An attribute-based infrastructure for EHR systems, where each patient's EHR files are encrypted using a broadcast variant of CP-ABE that allows direct revocation. However, the cipher text length grows linearly with the number of unrevoked users. In a variant of ABE that allows delegation of access rights is proposed for encrypted EHRs applied cipher text policy ABE (CP-ABE) to manage the sharing of PHRs, and introduced the concept of social/professional domains investigated using ABE to generate self-protecting EMRs, which can either be stored on cloud servers or cell phones so that EMR could be accessed when the health provider is offline.

Setup and Key Distribution

In this module the system first defines a common universe of data attributes shared by every PSD, such as "basic profile", "medical history", "allergies", and "prescriptions". An emergency attribute is also defined for break-glass access.

Each PHR owner's client application generates its corresponding public/master keys. The public keys can be

published via user's profile in an online healthcare social-network (HSN)

There are two ways for distributing secret keys.

First, when first using the PHR service, a PHR owner can specify the access privilege of a data reader in her PSD, and let her application generate and distribute corresponding key to the latter, in a way resembling invitations in GoogleDoc.

Second, a reader in PSD could obtain the secret key by sending a request (indicating which types of files she wants to access) to the PHR owner via HSN, and the owner will grant her a subset of requested data types. Based on that, the policy engine of the application automatically derives an access structure, and runs keygen of KP-ABE to generate the user secret key that embeds her access structure.

Break-glass module

In this module when an emergency happens, the regular access policies may no longer be applicable. To handle this situation, break-glass access is needed to access the victim's PHR. In our framework, each owner's PHR's access right is also delegated to an emergency department ED to prevent from abuse of break-glass option, the emergency staff needs to contact the ED to verify her identity and the emergency situation, and obtain temporary read keys. After the emergency is over, the patient can revoke the emergent access via the ED.

RESULTS & CONCLUSION

In this research paper, we have proposed a novel framework of secure sharing of personal health records in cloud computing. Considering partially trustworthy cloud servers, we argue that to fully realize the patient-centric concept, patients shall have complete control of their own privacy through encrypting their PHR files to allow fine-grained access. The framework addresses the unique challenges brought by multiple PHR owners and users, in that we greatly reduce the complexity of key management while enhance the privacy guarantees compared with previous works. We utilize ABE to encrypt the PHR data, so that patients can allow access not only by personal users, but also various users from public domains with different professional roles, qualifications and affiliations. Furthermore, we enhance an existing MA-ABE scheme to handle efficient and on-demand user revocation, and prove its security. Through implementation and simulation, we show that our solution is both scalable and efficient.

REFERENCES

1. User Interfaces in C#: Windows Forms and Custom Controls by Matthew MacDonald.
2. Applied Microsoft® .NET Framework Programming (Pro-Developer) by Jeffrey Richter.
3. Practical .Net2 and C#2: Harness the Platform, the Language, and the Framework by Patrick Smacchia.
4. Data Communications and Networking, by Behrouz A Forouzan.
5. Computer Networking: A Top-Down Approach, by James F. Kurose.
6. Operating System Concepts, by Abraham Silberschatz.
7. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," University of California, Berkeley, Tech. Rep. USB-EECS-2009-28, Feb 2009.
8. "The apache cassandra project," http://cassandra.apache.org/.
9. L. Lamport, "The part-time parliament," ACM Transactions on Computer Systems, vol. 16, pp. 133–169, 1998.
10. N. Bonvin, T. G. Papaioannou, and K. Aberer, "Cost-efficient and differentiated data availability guarantees in data clouds," in Proc. of the ICDE, Long Beach, CA, USA, 2010.
11. O. Regev and N. Nisan, "The popcorn market. online markets for computational resources," Decision Support Systems, vol. 28, no. 1-2, pp. 177 – 189, 2000.
12. A. Helsinger and T. Wright, "Cougaar: A robust configurable multi agent platform," in Proc. of the IEEE Aerospace Conference, 2005.
13. J. Brunelle, P. Hurst, J. Huth, L. Kang, C. Ng, D. C. Parkes, M. Seltzer, J. Shank, and S. Youssef, "Egg: an extensible and economics-inspired open grid computing platform," in Proc. of the GECON, Singapore, May 2006.
14. J. Norris, K. Coleman, A. Fox, and G. Candea, "Oncall: Defeating spikes with a free-market application cluster," in Proc. of the International

Conference on Autonomic Computing, New York, NY, USA, May 2004.

15. C. Pautasso, T. Heinis, and G. Alonso, "Autonomic resource provisioning for software business processes," Information and Software Technology, vol. 49, pp. 65–80, 2007.

16. A. Dan, D. Davis, R. Kearney, A. Keller, R. King, D. Kuebler, H. Ludwig, M. Polan, M. Spreitzer, and A. Youssef, "Web services on demand: Wsla-driven automated management," IBM Syst. J., vol. 43, no. 1, pp. 136–158, 2004.