1-1-2006

A unified framework for trust management

Weiliang Zhao
University of Western Sydney, wzhao@uow.edu.au

Vijay Varadharajan
Macquarie University

George Bryan
University of Western Sydney

A Unified Framework for Trust Management

Weiliang Zhao
School of Computing and Mathematics
University of Western Sydney
Email: wzhao@scm.uws.edu.au

Vijay Varadharajan
Department of Computing
Macquarie University
Email: vijay@ics.mq.edu.au

George Bryan
School of Computing and Mathematics
University of Western Sydney
Email: g.bryan@scm.uws.edu.au

Abstract— **In this paper, we propose a unified framework for trust management that can cover a broad variety of trust mechanisms including reputations, credentials, local data and environment parameters. The proposed trust management framework will leverage established standards and it covers a broad variety of situations in different environments. This framework can provide utilizing and enabling tools for trust management. Under this framework, different trust mechanisms can be assembled together when multiple mechanisms of trust are necessary. Here, we refer to our trust management system as TrustEngine. TrustEngine follows the initial ideas of PolicyMaker to separate generic mechanisms of trust management from application-specific policies which are defined by each application. TrustEngine has a generic set of functions, interfaces, and data storage for trust management in distributed environments. TrustEngine is an open system and it can easily include new trust components. We describe the architecture and implementation details of TrustEngine. We provide an application scenario to illustrate the usage of TrustEngine in the real world. We believe that the development of trust management in real applications can be automated to substantially higher level based on our proposed framework.**

I. Introduction

Trust management is an important issue in security analysis and design, particularly when centrally managed security is not possible. There are many services and applications that must accommodate appropriate notions of trust and related elements of trust such as community reputation and security credentials. Reputation-based systems such as XREP [1], NICE [2], P-Grid [3] provide facility to compute the reputation of an involved entity by aggregating the perception of other entities in the system. Some reputation systems like TrustNet [4] and NodeRanking [5] utilize existing social relationships to compute reputations based on various parameters. M. Kinateder et al [6] proposed a generic model for trust based on reputation. Normally, these systems are limited in the sense that they do not link the purpose of reputation to the evaluation and they employ reputation as the exclusive evidence for trust. For example, the reputation of an entity will determine the restriction of access to resources and services. On the other hand, there are many trust management systems based on credentials exclusively. Public key certificates X.509 and PGP have already used credentials to deal with trust management problem. As a further step, M. Blaze et al firstly identified trust management as a distinct and important component of security in distributed environments and proposed PolicyMaker [7]. After that, several automated trust management systems have been proposed and implemented including PolicyMaker [7], KeyNote [8] and REFEREE [9]. All the above systems use credentials as evidence of required trust. Normally, there are credential verification and secure application policies to restrict access to resources and services. G. Suryanarayana, et al [10] pointed out that these systems are limited in the sense that they do not enable an entity to aggregate the perception of other entities in the system in order to choose a suitable reputable service.

All existing trust management systems focus on building up a new trust management layer and the concept of trust is always assumed in a specific context. In the computing world, trust has been studied in multiple dimensions. Trust had been introduced as a security concept in trusted systems [11] and trusted computing [12]. S. Marsh had formalized trust as a computational concept [13]. In order to have a solid understanding of the concept of trust relationship, we have proposed a strict definition for trust relationship and have developed a set of tools to model the trust relationships in distributed environments [14]. The target of the formal definition of trust relationships is not only to reflect many of the commonly used notions of trust but also to provide a taxonomy framework where a range of useful trust relationships can be expressed and compared. Following the above work, we have discussed different properties of trust in distributed information systems and considered the development of an overall methodology targeted at capturing the life cycle of trust relationships [15]. Our previous works form the basis of the unified framework for trust management proposed in this paper.

Majority of the current solutions on trust management are only suitable for dealing with static forms of trust. However, trust can change depending on real time situations. Hence the dynamic properties of trust must be included in trust management systems. Currently, there are two approaches for establishing trust between involved parties. One approach assumes that the involved parties are not strangers and can present a local identity to obtain the predefined trust target. Another approach assumes that the involved parties are strangers and they negotiate trust based on the iterative exchange of digital credentials to establish one-way or mutual trust. The trust negotiation is based on cryptographically signed credentials [16], [17], [18]. Credentials can be used as evidences or testimonials for one's right to credit, for achieving a certain level of confidence, or for authority in trust management. A predominant approach to trust has been based on authenti-

1-4244-0423-1/06/$20.00 ©2006 IEEE

cation of identities; another approach to trust has been the reliance on the use of credentials (this can involve multiple credentials and a complex negotiation process). However, both the identity based and credential based approaches can be inadequate in dynamic distributed environments, where there is often no centralized control and the trust characteristics are prone to changes over time. These require some context based mechanisms for trust management.

In this paper, we propose a trust management framework that can address the above mentioned limitations of current trust management systems. The framework covers a broad variety of trust mechanisms including reputations and credentials. Our framework will leverage established standards of trust management and it covers a range of situations in different environments. The main aim of the framework is to establish infrastructure and tools of trust management for developing distributed applications. Different trust mechanisms can be assembled together as necessary for the applications concerned.

The rest of the paper is structured as follows. In section II, we provide some general discussions about the proposed trust management framework. In section III, we propose our trust management architecture. In section IV, we discuss the implementation details of the proposed architecture. In section V, we consider an application example illustrating the use of the proposed framework in practice. In section VI, we provide some concluding remarks.

II. Trust Management Framework

Our unified framework of trust management leverages established standards of trust management and covers a broad variety of situations in different environments. The main aim of the framework is to establish infrastructure and tools of trust management for developing distributed applications. The framework provides the generic architecture and implementation guidelines for trust management in distributed information systems. The tasks of developing trust management system can be automated to a substantial degree with the help of the proposed trust management framework. Our previous works about formal definition of trust relationships and the properties of trust [14], [15] provide the general understanding of trust and they form the basis of the proposed trust management framework. We will follow the basic idea of M. Blaze et al [7] to deal with trust management as an independent layer in distributed applications. In this paper, we will refer to our trust management system as a TrustEngine. TrustEngine follows the initial ideas of PolicyMaker, which are to separate generic mechanisms of trust management from application-specific policies that are defined by each application. Beyond all credential-based systems, TrustEngine will cover other trust mechanisms such as reputation in the same unified framework. TrustEngine has a collection of functions, interfaces and data storage. We design TrustEngine as an open system thereby enabling different independent trust components to be included.

The trust management framework will be based on the following general principles:

Unified Framework: Existing systems treat different trust mechanisms separately. There is no framework to handle trust related issues in a comprehensive and consistent manner. Our approach allows multiple established trust mechanisms to cooperate with each other under the same unified framework.
Flexibility: Our trust management framework can support complex situations of trust management. The framework covers commonly used notions of trust and provides a taxonomy allowing a range of useful trust relationships to be expressed and compared.
Locality of Control: The entity involved in the trust relationship can make trust decisions based on the specific circumstances. The framework supports local control of trust relationships. The local control can based on different trust parameters such as credentials, community-based reputation, data from storage and environment conditions.
Standard Interfaces: TrustEngine has standard interfaces to applications or application-specific policies. The interfaces provide unified mechanisms for requesting trust, evaluating trust and consuming trust (consuming trust will be described in section III-E).

III. TrustEngine Architecture

An important objective of the trust management architecture is to support a wide range of different context-based trust policies. The trust policies are normally task-specific and they may be supported by multiple mechanisms. A trust policy can require one or multiple trust relationships. When a trust policy is enforced, the required trust relationships in the trust policy must be satisfied. The trust decision is context-based and it is based on the evaluation of one or multiple trust relationships. In this paper, we propose a TrustEngine to hold all trust related components that could be separated from applications. The formal definition of a trust relationship is the starting point for the trust management architecture. TrustEngine addresses applications' trust requests like a database query engine. TrustEngine accepts as inputs a trustor, a trustee, a set of conditions, and a set of properties that describe the actions or attributes of the trustees. Depending on the form of the query, TrustEngine can return yes/no answer or additional restrictions that would make the trust evaluation possible. Trust relationships are specified and implemented in the TrustEngine when applications are being developed. At runtime, related trust relationships are located, evaluated and consumed.

TrustEngine is a container for all trust components and it has the flexibility to be expanded easily to hold new trust components. Each component in TrustEngine performs some trust function or has some data storage to be used by other trust functions. TrustEngine includes TrustDatabase for storage of trust related data and component packages such as LocatingTrust, EvaluatingTrust and ConsumingTrust. See Fig.1.

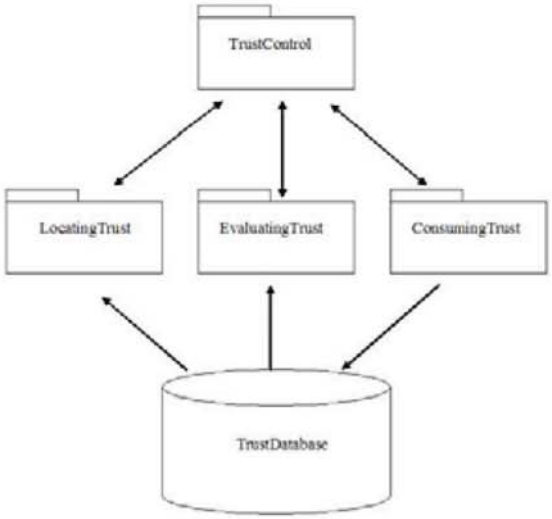

Fig. 1. TrustEngine Package Hierarchy

A. TrustDatabase

TrustEngine includes a persistent storage mechanism for storing and retrieving information about trust. TrustDatabase is the data storage of TrustEngine that maintains trust relationships. The storage mechanism can be a relational database or data profile. When an application is being developed, trust relationships in the application must be analyzed and modelled. These trust relationships must be loaded into the TrustDatabase before the application is running.

B. TrustControl

TrustControl is the package for the overall management and control of TrustEngine at run time. TrustEngine controller is the only computing component in this package. It is the general controller of TrustEngine which links applications and functional packages of TrustEngine (LocatingTrust, EvaluatingTrust and ConsumingTrust). See Fig.2.

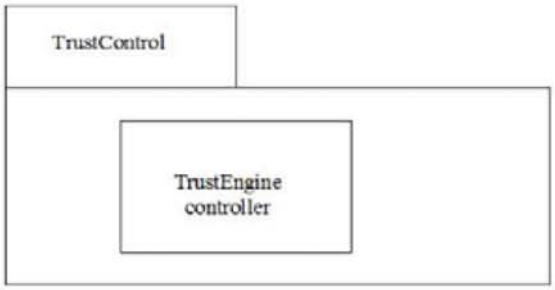

Fig. 2. Components of TrustControl

C. LocatingTrust

LocatingTrust is the package for finding the trust relationship based on the request. There are three components in this package which are referred to as locating trust controller, trust relationship locator and authentication controller. Locating trust controller is the management component which receives the request from applications and it assigns tasks to trust relationship locator and authentication controller. Trust relationship locator is the component to find the requested trust relationship from the TrustDatabase. Authentication controller is the component which deals with authentication and provides an interface to the authentication service in the system. See Fig.3.

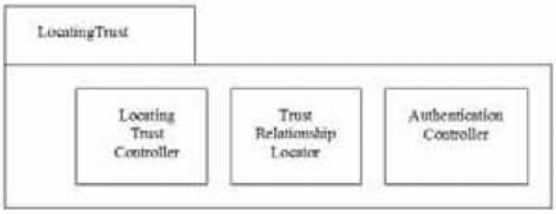

Fig. 3. Components of LocatingTrust

D. EvaluatingTrust

EvaluatingTrust contains computing components required for the evaluation of a trust relationship. The evaluation of a trust relationship involves checking whether the conditions of a trust relationship can be satisfied or not. The conditions of trust take into account the risks from evil actions of trustees, evil actions from other parties, and from unstable environments. There are different conditions in a trust relationship and there may be multiple mechanisms to support trust evaluation. The existing reputation-based systems and credential based systems can be employed as components of trust evaluation. Any successful system or mechanism for checking or evaluating the evidence and reputation can be included in the EvaluatingTrust package. Hence existing standards and successful systems related to trust can be employed by this unified trust management framework easily. In EvaluatingTrust, trust evaluation controller is the computing component that assigns the evaluation tasks to other functional components in this package. EvaluatingTrust has functional components for specific evaluating tasks such as credential evaluation, reputation evaluation, stored data evaluation, and environment evaluation. In the implementation, the package of EvaluatingTrust will be customized based on the specific requirements. See Fig.4.

E. ConsumingTrust

ConsumingTrust contains the computing components for consuming trust. Consuming trust deals with how to use the output of the evaluation of a trust relationship. ConsumingTrust contains consuming controller and two next level packages ApplicationConsuming and SystemConsuming. Consuming controller is the manager of trust consuming. It receives the result of trust evaluation and assigns consuming tasks to ApplicationConsuming and SystemConsuming. ApplicationConsuming deals with the consuming of trust by

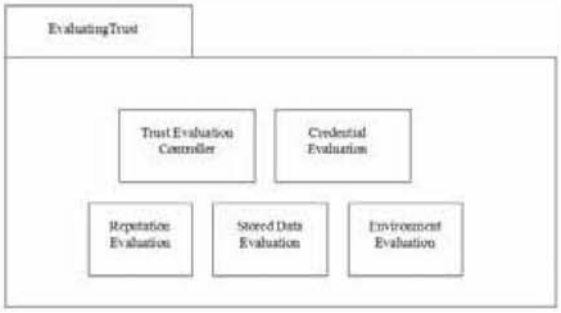

Fig. 4. Components of EvaluatingTrust

applications. SystemConsuming deals with the consuming of trust by trustEngine and auditing system. See Fig.5.

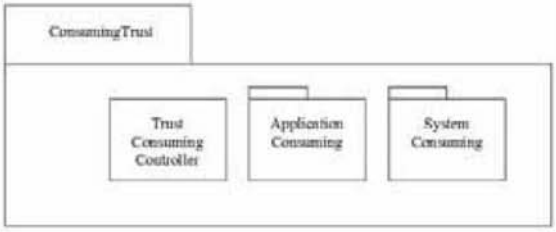

Fig. 5. Components of ConsumingTrust

1) ApplicationConsuming:: In application consuming, the evaluation of trust relationship is not always be consumed immediately. The result of evaluation of trust relationship can be stored and distributed in different ways. There are three normal ways to use the output of trust evaluation. The first way is that the result of trust evaluation is immediately used by consuming applications. The computing component for this way is direct trust consuming controller. The second way is to generate credentials with the result of trust evaluation as input. These credentials will be used in the future by the same or other applications. Credential generator controller is the corresponding computing component. The third way is that the result of trust evaluation is stored in database and the data will be retrieved and used by applications in the future. Consuming data storage controller is the corresponding computing component. Application consuming controller plays the role of manager for application consuming. Application consuming controller receives tasks from consuming controller and it assigns tasks to direct trust consuming controller, credential generator controller and consuming data storage controller. See Fig.6.

2) SystemConsuming:: The system consuming has three components which are system consuming controller, TrustEngine consuming controller and auditing consuming controller. System consuming controller receives tasks from consuming controller and it assigns tasks to TrustEngine consuming controller and auditing consuming controller. TrustEngine consuming controller deals with the consuming

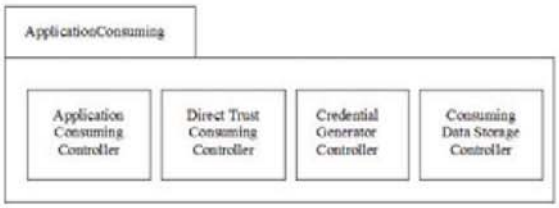

Fig. 6. Components of ApplicationConsuming

of trust by TrustEngine. Auditing consuming controller deals with the consuming of trust by auditing system.

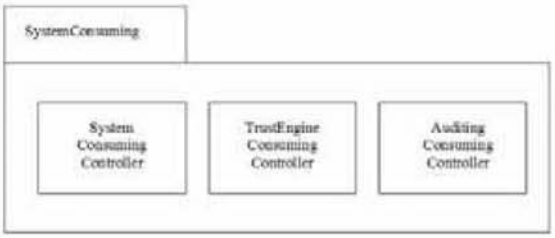

Fig. 7. Components of SystemConsuming

IV. Implementation of TrustEngine

Here we provide some details of implementation of TrustEngine architecture described in last section. At first, we will describe the system components one by one. The description will focus on the functions and interfaces of the system components. Then we provide a sequence of system operations. These operations are generic.

A. System Components

TrustEngine Controller: TrustEngine controller is a runtime controller of TrustEngine. It has an interface to receive trust request from applications and an interface to return necessary information feedback from TrustEngine to request applications. It assigns tasks to locating trust controller, trust evaluating controller and trust consuming controller. It receives return information from locating trust controller, trust evaluating controller and trust consuming controller. It performs management tasks for TrustEngine among computing packages LocatingTrust, EvaluatingTrust and ConsumingTrust.

Locating Trust Controller: Locating trust controller has an interface to receive requests from TrustEngine controller. It forwards the request to trust relationship locator for finding the related trust relationship. The locating trust controller has an interface to return the status of locating of the required trust relationship to TrustEngine controller. The locating trust controller assigns authentication controller to do the task of authentication for the involved trustee. It has an interface to return information of trust locating to TrustEngine controller.

Trust Relationship Locator: Trust relationship locator performs function to find the related trust relationship. It has an interface in connection with TrustDatabase where trust

relationships are maintained. It has another interface to return the searching result to locating trust controller.

Authentication Controller: Authentication controller performs function to authenticate trustee in required trust relationship. It has interface to receive authentication task from locating trust controller and it has an interface to leverage existing standards for the authentication. It has an interface to return authentication information (authentication tokens or status) to locating trust controller.

Trust Evaluation Controller: Trust evaluation controller performs function of the management of trust evaluation. It has an interface to receive tasks of trust evaluation from TrustEngine controller. It assigns evaluation tasks to computing components credential evaluation, reputation evaluation, stored data evaluation and environment evaluation. It has an interface to return evaluating result to TrustEngine controller.

Credential Evaluation: Credential evaluation is the computing component for credential evaluation. It includes multiple evaluating mechanisms for different credentials. It has computing functions and/or provides interfaces linking with existing computing utility of credential evaluation. It has an interface to receive tasks from trust evaluation controller and an interface to return the result of evaluation to trust evaluation controller.

Reputation Evaluation: Reputation evaluation looks after computing tasks about reputation evaluation. It includes computing functions for reputation calculation and/or interfaces to existing utility of reputation evaluation. It has an interface to receive task from trust evaluation controller and an interface to return the result of evaluation to trust evaluation controller.

Stored Data Evaluation: Stored data evaluation looks after the evaluation of trust against stored data. It has an interface to receive tasks from trust evaluation controller and an interface to return the result of evaluation to trust evaluation controller.

Environment Evaluation: Environment evaluation looks after the evaluation of trust against environment variables. It has an interface to receive tasks from trust evaluation controller and an interface to return the result of evaluation to trust evaluation controller.

Trust Consuming Controller: Trust consuming controller performs the management of trust consuming of TrustEngine. It has an interface to receive the consuming tasks from TrustEngine controller and an interface to return the consuming result/status to TrustEngine controller. It assigns consuming tasks to application consuming controller and system consuming controller.

Application Consuming Controller: Application consuming controller performs the management of consuming of trust by applications. It has an interface to receive tasks from trust consuming controller and an interface to return the consuming result/status to trust consuming controller. It assigns tasks to direct trust consuming controller, credential generator controller and consuming data storage controller.

Direct Trust Consuming Controller: Direct trust consuming controller looks after the consuming of trust when the evaluation of trust relationship is consumed immediately by the request application. It has an interface to receive the consuming tasks from the application consuming controller and an interface to return consuming status to application consuming controller. It has an interface for consuming of trust relationship by the request application.

Credential Generator Controller: Credential generator controller looks after the consuming of trust when the evaluation of trust relationship is consumed by generation of credentials. It has an interface to receive the consuming tasks from the application consuming controller and an interface to return consuming status to application consuming controller. It has functions to generate and manage credentials. Existing standards and computing utility can be employed in the generation and management of the credentials.

Consuming Data Storage Controller: Consuming data storage controller looks after the consuming of trust when the evaluation of trust relationship is consumed by storing related information in database. The information stored in database will be retrieved by applications in the future. It has an interface to receive the consuming tasks from the application consuming controller and an interface to return consuming status to application consuming controller. It has functions to format data and has interfaces to save data with different data storage mechanisms such as local database, remote database and profiles.

System Consuming Controller: System consuming controller performs the management of consuming of trust by system. It has an interface to receive tasks from trust consuming controller and an interface to return the consuming result/status to trust consuming controller. It assigns tasks to TrustEngine consuming controller and auditing consuming controller.

TrustEngine Consuming Controller: TrustEngine consuming controller looks after the consuming of trust by TrustEngine itself. It has an interface to receive the consuming tasks from the TrustEngine consuming controller and an interface to return consuming status to TrustEngine consuming controller. It has functions for trust consuming by TrustEngine and interfaces to save data in TrustDatabase.

Auditing Consuming Controller: Auditing consuming controller looks after the consuming of trust for the auditing purpose. It has an interface to receive the consuming tasks from the TrustEngine consuming controller and an interface to return consuming status to TrustEngine consuming controller. It has interfaces to link to auditing functions or database in the system.

B. System Setting Up and Operations

In this sub section, we provide a generic description of system setting up and system operations. In the development of trust management system, the system components described in last sub section will be customized based on specific requirements of the distributed information system. The required computing components will be installed. At runtime, a set of operations of these components will be activated based on the specific trust request from applications.

In package TrustControl, TrustEngine controller is the only computing component and it links applications and computing

components in packages of LocatingTrust, EvaluatingTrust and ConsumingTrust. In package LocatingTrust, there are computing components as locating trust controller, trust relationship locator and authentication controller. In package EvaluatingTrust, there are computing components as trust evaluation controller, credential evaluation, reputation evaluation, stored data evaluation and environment evaluation. In package ConsumingTrust, there are computing components as trust consuming controller, application consuming controller, direct trust consuming controller, credential generator controller, consuming data storage controller, system consuming controller, TrustEngine consuming controller and auditing consuming controller.

When there is trust request from applications, system operations of TrustEngine will be activated in the following sequence:

TC1: TrustEngine controller is the first computing component to be activated and it will assign a task to locating trust controller.
LT1: Locating trust controller assigns a task to trust relationship locator.
LT2: Trust relationship locator finds the required trust relationship and return it to locating trust controller.
LT3: Locating trust controller requires authentication controller to do the task of authentication for the involved trustee.
LT4: Authentication controller performs the task of authentication.
LT5: Locating trust controller returns information of locating of trust to TrustEngine controller.
TC2: TrustEngine controller requires trust evaluation controller to evaluate the trust relationship.
TE1: Trust evaluation controller assigns evaluation tasks to computing components credential evaluation, reputation evaluation, stored data evaluation and environment evaluation.
TE2: Credential evaluation checks credentials.
TE3: Reputation evaluation performs the computing tasks of reputation evaluating.
TE4: Stored data evaluation performs the evaluation of trust against stored data.
TE5: Environment evaluation performs the evaluation of trust against environment variables.
TE6: Trust evaluation controller integrates the results of TE2, TE3, TE4, and TE5 and returns final evaluating result to TrustEngine controller.
TC3: TrustEngine controller assigns trust consuming controller to manage the consuming of the evaluated trust relationship.
TU1: Trust consuming controller assigns consuming tasks to application consuming controller and system consuming controller.
TUA1: Application consuming controller assigns tasks to direct trust consuming controller, credential generator controller and consuming data storage controller.
TUA2: Direct Trust Consuming Controller informs the application who initiates the trust request for the consuming of trust.
TUA3: Credential generator controller generates credentials based on the result of trust evaluation. The credentials will be stored or delivered based the specific requirements of a real system.
TUA4: Consuming data storage controller formats data and saves them with different data storage mechanisms such as local database, remote database and profiles.
TUS1: System consuming controller assigns tasks to TrustEngine consuming controller and auditing consuming controller.
TUS2: TrustEngine consuming controller performs functions for trust consuming by TrustEngine and it saves data in TrustDatabase.
TUS3: Auditing consuming controller performs functions of trust consuming for the auditing purpose.

The real trust management system must be developed based on specific business requirements, available technologies and computing environments. The business requirements of trust are expressed by trust relationships. These trust relationships play a crucial role in trust management. How to model these trust relationships can be found in our previous work [14], [15]. The computing components of TrustEngine described above will be customized based on these trust relationships. These generic computing components of TrustEngine must be transformed into solid computing components coupled with available technologies and computing environments. The existing standards and conventions are leveraged in such a process.

V. An Application Example

In order to illustrate our TrustEngine architecture proposed above, we make a scenario example based on possible requirements in the federated medical services. In federated distributed medical services, there are multiple trust relationships between entities such as patients, physicians, hospitals, insurance companies, pharmacies, etc and we believe that trust plays an important role. The modelling and evaluating of trust relationships are beyond the normal authentication and authorisation. Trust relationships are context-based and must be evaluated dynamically. Trust relationships may be modified at any time. We will employ TrustEngine architecture described in section III and section IV to develop the sub system for trust management.

In such a system, there are many trust relationships and it could be very complicated, but we only consider some of them for illustrating our TrustEngine architecture. In federated medical services, there are an enormous variety of applications that require making complex trust decisions that are dependent on runtime situations. The trust requirements are normally dynamic and flexible. Trust mechanism in federated medical services needs to be highly dynamic and independent from any particular application. Here we will choose three typical trust relationships in the federated medical services and use them as examples to discuss the evaluation and consuming of trust in a real system. We provide some discussions about the system

setting up for trust management in federated medical services. Then we give two run time scenarios based on corresponding trust relationships. We hope that readers can get a general feeling of TrustEngine architecture and the framework for trust management.

A. Modelling Trust Relationships

In our previous work [14], [15], we have discussed how to model trust relationship in distributed information systems based on proposed formal definition of trust relationship and properties of trust relationships. There are several stages for modelling trust relationships in distributed information systems such as extracting trust requirements, identifying possible trust relationships from trust requirements, choosing the whole set of trust relationships from possible trust relationships and implementing and maintaining trust relationships. The trust relationships in federated medical services are very complicated. We will not consider the details of trust relationships in such a system. For our purpose, here we only model the following three trust relationships to illustrate the usage of TrustEngine architecture. The trust relationships are:

1. $T1 = < R1, E1, C1, P1 >$. $R1$ includes patients; $E1$ includes doctors; $C1$ includes medical practitioner licences; and $P1$ includes that doctors have the ability to do general practice.
2. $T2 = < R2, E2, C2, P2 >$. $R2$ includes patients; $E2$ includes doctors; $C2$ includes cardiologist licences; and $P2$ includes that doctors have the ability to do heart checks or attend the heart surgeries as non-principal doctor.
3. $T3 = < R3, E3, C3, P3 >$. $R3$ includes patients; $E3$ includes doctors; $C3$ includes cardiologist licences, reputation for more than 5 year cardiology practice, experience of successful heart surgery in the specified hospital and there is surgery room in specified date and hospital; $P3$ includes that doctors have the ability to be principal doctor in the heart surgery at the specified hospital on a specified date.

These trust relationships are stored in TrustDatabase before they may be used by any other computing component in TrustEngine.

B. System Setting Up

The sub system for trust management of federated medical services will utilize TrustEngine architecture described in III and IV to perform all computing tasks about trust. We use the federated medical services as an example to cover all the computing components in TrustEngine architecture. The computing components in packages TrustControl and LocatingTrust are always necessary in any system. Different authentication mechanisms can be employed using the interface of authentication controller. In the package of TrustEvaluating, the computing components will be customized according to specified requirements. In federated medical services, it is possible to evaluate trust against credential, reputation, stored data and environment parameters and therefore all computing components in package TrustEvaluating should be installed. In federated medical services, all the three application consuming ways may be involved. The direct trust consuming controller, credential generator controller and consuming data storage controller are all necessary to be developed and installed in the system. TrustEngine consuming controller is installed for the result of trust evaluation to be used by TrustEngine. Auditing consuming controller is installed for the result of trust evaluation to be used by auditing system.

C. Run Time Scenarios

Here we provide two run time scenarios based on the corresponding trust relationships modelled in V-A. We assume that the whole system has been set up and all necessary computing components have been installed. In these scenarios, we will provide the sequence of operations at run time. We hope these scenarios are helpful for readers to understand the computing components and operations of TrustEngine.

Scenario 1: When a patient books a general medical practice through federated medical services, trust relationship $T1$ in section V-A will be involved. The request of trust is initiated by booking application of federated medical services. At run time, system operations will be activated in the following orders: TC1, LT1, LT2, LT3, LT4, LT5, TC2, TE1, TE2, TE6, TC3, TU1, TUA1, TUA2, TUS1, TUS2, TUS3. These operations perform whole set of trust management tasks for the involved trust request. Particularly, TE2 is the operation to verify the validity of the medical practitioner licence associated with the involved doctor. We assume that booking application will use the evaluated trust relationship immediately and TUA2 is the operation for the direct trust consuming. TUS1, TUS2, TUS3 are operations for system consuming based on specific system requirements.

Scenario 2: When a patient books a heart surgery through federated medical services, trust relationship $T3$ in section V-A will be involved. This trust relationship is complicated and it needs multiple mechanisms for the evaluation. There are multiple ways for the consuming of this trust relationship as well. We assume that the request of trust is initiated by the booking application of surgeries and trust management is controlled by information system of the specified hospital of the possible surgery. At run time, when the trust request is sent to the information system of specified hospital, system operations will be activated in the following orders: TC1, LT1, LT2, LT3, LT4, LT5, TC2, TE1, TE2, TE3, TE4, TE5, TE6, TC3, TU1, TUA1, TUA2, TUA3, TUA4, TUS1, TUS2, TUS3. For trust evaluation, TE2 verifies the cardiologist licence; TE3 calculates and checks over all reputation of the doctor over recent 5 years; TE4 checks the experience of successful heart surgery in the specified hospital; TE5 checks there is surgery room or not in specified date and hospital. TE6 will integrate the results of TE2, TE3, TE4 and TE5 and return the overall result to TrustEngine controller. In this scenario, we assume that the evaluated trust will be used by the booking application of heart surgery. Based on the trust evaluation, some credentials (certificates) can be generated

to provide the information about this evaluated trust and the credentials will be delivered for further usage in the system. The evaluated trust will be also stored in database for further usage of applications in the information system. TU1, TUA1, TUA2, TUA3, TUA4 are activated one by one. TUS1, TUS2, TUS3 are possible operations for system consuming.

VI. Concluding Remarks

In this paper, we have proposed a unified framework for trust management. The proposed framework and architecture make it possible for developers to create a trust management system by simply implementing some business logic. The development of trust management in real applications can be automated to substantially higher level based on the proposed framework. The framework can cover multiple trust mechanisms and computing components of trust can be easily assembled. Particularly, the existing reputation-based trust systems and credential-based trust systems are put in a unified framework. The trust request, trust evaluation and trust consuming are handled in a comprehensive and consistent manner. The unified framework provides multiple interfaces and they can be implemented based on specific requirements in the real world and existing standards can easily be leveraged.

References

[1] E. Damiani, D. C. d. Vimercati, S. Paraboschi, P. Samarati, and F. Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks," in *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*. ACM Press, 2002, pp. 207–216, washington, DC, USA.

[2] S. Lee, R. Sherwood, and B. Bhattacharjee, "Cooperative peer groups in nice," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, vol. 2, 2003, pp. 1272–1282 vol.2, tY - CONF.

[3] K. Aberer and Z. Despotovic, "Managing trust in a peer-2-peer information system," in *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*. ACM Press, 2001, pp. 310–317, atlanta, Georgia, USA.

[4] M. Schillo, M. Rovatsos, and P. Funk, "Using trust for detecting deceitful agents in artificial societies," *Applied Artificial Intelligence Journal, Special Issue edited by Castelfranchi, C., Tan, Y., Falcone, R. and Firozabadi, B. on Deception, Fraud and Trust in Agent Societies.*, vol. 14, no. 8, pp. 825–848, 2000.

[5] J. M. Pujol, R. Sang, esa, and J. Delgado, "Extracting reputation in multi agent systems by means of social network topology," in *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*. ACM Press, 2002, pp. 467–474, bologna, Italy.

[6] M. Kinateder, E. Baschny, and K. Rothermel, "Towards a generic trust model - comparison of various trust update algorithms," vol. 3477, pp. 177–192, 2005.

[7] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *Proceedings of IEEE Symposium on Security and Privacy*, 1996, pp. 164–173.

[8] M. Blaze, J. Feigenbaum, and A. Keromytis, "KeyNote: Trust management for public-key infrastructures (position paper)," *Lecture Notes in Computer Science*, vol. 1550, pp. 59–63, 1999.

[9] Y. H. Chu, J. Feigenbaum, B. LaMacchia, P. Resnick, and M. Strauss, "REFEREE: Trust management for Web applications," *Computer Networks and ISDN Systems*, vol. 29, no. 8–13, pp. 953–964, 1997.

[10] G. Suryanarayana, J. Erenkrantz, S. Hendrickson, and R. Taylor, "Pace: an architectural style for trust management in decentralized applications," in *Software Architecture, 2004. WICSA 2004. Proceedings. Fourth Working IEEE/IFIP Conference on*, 2004, pp. 221–230, tY - CONF.

[11] TCSEC, "Trusted computer system evaluation criteria," U.S.A National Computer Security Council, Tech. Rep., 1985, dOD standard 5200.28-STD.

[12] J. Landauer, T. Redmond, and T. Benzel, "Formal policies for trusted processes," in *Proceedings of the Computer Security Foundations Workshop II, 1989*, 1989, pp. 31–40.

[13] S. Marsh, "Formalising trust as a computational concept," PHD thesis, University of Sterling, 1994.

[14] W. Zhao, V. Varadharajan, and G. Bryan, "Modelling trust relationships in distributed environments," *Lecture Notes in Computer Science*, vol. 3184, pp. 40–49, 2004.

[15] W. Zhao, V. Varadharajan, and G. Bryan, "Type and scope of trust relationships in collaborative interactions in distributed environments," in *7th International Conference on Enterprise Information Systems*, vol. 3, Miami, Florida, 2005, pp. 331–336.

[16] M. N. Huhns and D. A. Buell, "Trusted autonomy," *Internet Computing, IEEE*, vol. 6, no. 3, pp. 92–95, 2002.

[17] W. H. Winsborough, K. E. Seamons, and et al, "Automated trust negotiation," in *Proceedings of DARPA Information Survivability Conference and Exposition*, 2000.

[18] M. Winslett et al, "Negotiating trust in the web," *IEEE Internet Computing*, vol. 6, no. 6, pp. 30–37, 2002.