A Blueprint for Universal Trust Management Services

Tomasz Kaszuba Krzysztof Rzadca Adam Wierzbicki
Grzegorz Wierzowiecki
Polish-Japanese Institute of Information Technology
Warsaw, Poland
adamw@pjwstk.edu.pl

November 20, 2008

Abstract

In the future, trust management may become yet another, standard service of information security, such as authentication, authorization, privacy or integrity. For this to happen, it is necessary to define standard primitives of trust management, and agree about what is in common among the many different applications of trust management studied to date. The *uTrust project* (Universal Trust) is an attempt to move towards the goal of creating standard trust management services. The scientific goal of the project is to define, formalize and implement universal trust management methods and verify their effectiveness. The practical goal of the project is to create a library of universal trust management methods, algorithms, and protocols. The trust management library will be developed as an Open Source project. We encourage the entire community to help with development, testing and evaluation. In this paper, a blueprint for the design of the uTrust library is presented, along with ideas for an evaluation of universal TM services.

1 Introduction

In open, distributed systems, autonomous users frequently have to make decisions under uncertainty. The outcomes of their decisions (and the welfare or utility of the users) are under risk, because they may depend on actions of other users, over which there is no centralized control. The goal of trust management is the support of decisions and actions of agents (users or programs) in an open, distributed system, under uncertainty and risk. To achieve this goal, trust management uses various kinds of information allowing users to determine what to expect about the behaviour of others. There exist many different definitions and formal models of concepts used in trust management, as well as many practical trust management methods.

In the future, trust management may become yet another, standard service of information security, such as authentication, authorization, privacy or integrity. For this to happen, it is necessary to define standard primitives of trust management, and agree about what is in common among the many different applications, frameworks, architectures and languages of trust management studied to date. Currently, although there are many practical trust management systems, they are applied in widely different domains: some of them have centralized components, others are fully distributed; some use reputation, while others rely on recommendations for transferring or delegating trust. This makes it difficult to propose a unified approach to managing trust.

The *uTrust project* (Universal Trust) is an attempt to move towards the goal of creating standard trust management services. The scientific goal of the project is to define, formalize and implement new, universal trust management methods and verify their effectiveness. The practical goal of the project is to create a **library of universal trust management methods, algorithms, and protocols**. This library will be distributed under public licence, and could also be made available as a Web Service. All researchers, developers, analysts interested in trust management are invited to participate in this effort. The trust management library will be developed as an Open Source project, and will be made available to the entire community for testing and evaluation.

In this paper, a blueprint for a library of universal Trust Management services is presented and discussed. **A challenge in the design of the library is too make it sufficiently general, yet not too abstract**. The next section discusses the interfaces of universal TM services. Section 3 presents an architecture of a library that could implement universal TM services. Section 4 considers the criteria for the evaluation of such a library. Section 5 concludes the paper.

2 Universal TM Services

When Trust Management (TM) is treated as a service, its interfaces must be sufficiently general to support a variety of applications. In this section (and on Figure 1), we present a generic scenario of an application that uses TM services. We introduce this scenario on two example applications: an Internet auction system and a Web service application. In the first case, the ”agent” depicted on Figure 1 is the auction system itself. In the second example, the agent is an entity that

invokes the Web service.
The scenario begins with an authentications phase. Nothing is assumed about the authentication mechanism, although its quality will have an impact on the effectiveness of various TM algorithms and protocols. In the case of the Internet auction system, authentication may be based on a simple pseudonymous login and password. In the case of Web service, there may be certificate-based authentication.

2.1 Universal Encounter Description

An agent that requests the help of TM service passes the description of the future interaction, called an *Encounter*. An Encounter [3] models all possible interactions between agents or applications that use TM services.

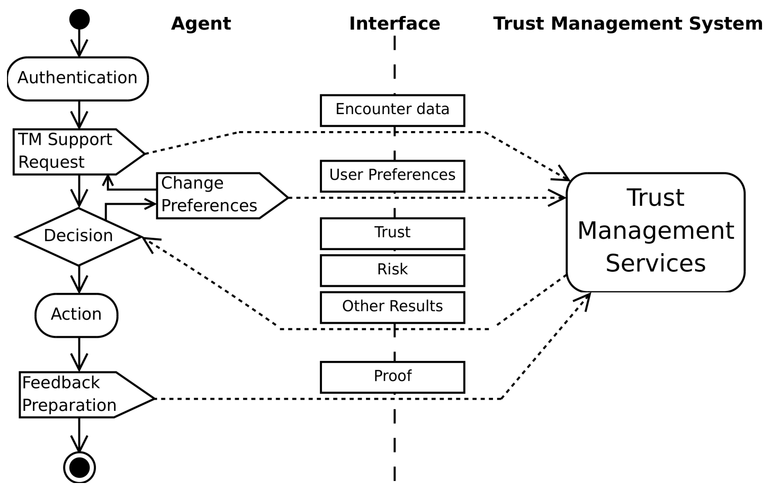

Figure 1: Universal TM Services and Interfaces

Relevant information about encounters should include agent identities, context, the actions available to agents and their outcomes.
In the case of an Internet auction, encounter data includes the identity of a user (buyer or seller), and possible actions: sending of purchased goods, or not sending them; paying or not paying the agreed price. For a Web service, encounter data includes an identity and the available actions are the following: returning the correct result of the invocation in reasonable time, returning incorrect results or delaying the reply. Context data for a Web service may include parameters of the Web service that could influence running times.

2.2 Results: Trust, Risk, Credibility and Others

TM service returns values of trust, risk, credibility or other information, depending on the type of invoked service. Note that the library is capable of incorporating various definitions of these concepts [4]. For example, in our research, we adopt the definition of *trust as a tolerance of risk*. This definition is close to that of dependability trust [2], but emphasizes that trust and risk can be values expressed on the same scale, thus enabling a direct comparison.
For the Internet auction, the TM service can return the reputation represented on a simple ordinal scale. In the case of the Web service, trust may be not represented at all. The TM service could return information about the policies that have been satisfied by available information. In addition, the TM service can return the risk that a service takes an excessively long time to return a result.

In the case of the Web service, trust may be not represented at all. The TM service could return information about the policies that have been satisfied by available information. In addition, the TM service can return the risk that a service takes an excessively long time to return a result.

2.3 Trust Management as Decision Support

We consider that TM services should be treated as a tool in decision support. The user must make a decision under conditions of uncertainty or risk. Viewing TM as decision support emphasizes the fact that the decision making process is interactive, and the user may influence TM services by specifying her preferences. Such an approach allows for a more universal specification of the TM services' interface. In the case of the Internet auction, the user could request an escrow service in the interactive stage, which changes the definition of the encounter and, consequently, influences the resulting trust.

2.4 Feedback by Universal Proofs

After the encounter, the agent passes feedback information to the TM system. To represent this information, the library uses the concept of a *Proof*, a universal representation of an encounter. In the Internet auction, a proof is a history-based report. In the Web service, a proof, called a ”security token”, delegates trust obtained in previous encounters with a trust management authority. Proofs can be even prior trust assumptions. Proofs can also be added at any other moment. In the Web service example, proofs could be presented by the agent at the beginning of the scenario, possibly following a trust negotiation procedure.

3 Internal Architecture of TM Services

A library of universal trust management services must be designed so that many different applications could use common primitives and data. It should be able to incorporate diverse methods, various algorithms and protocols of trust management. The challenge in the design of a library of universal TM services is therefore the **discovery of a common basis for the largest possible set of TM methods**.

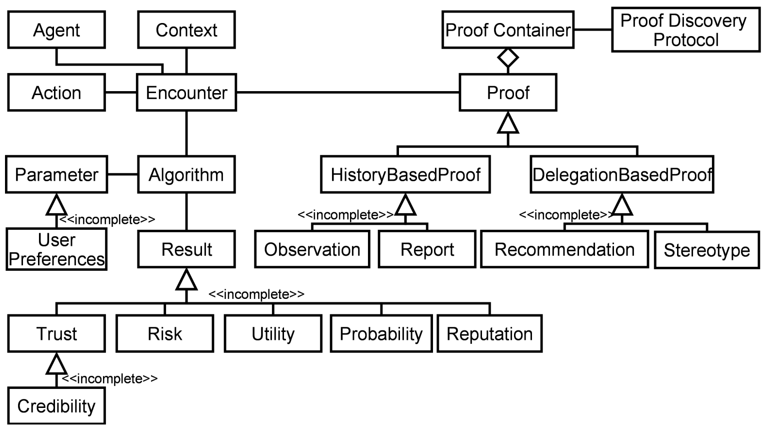

Figure 2: Class Diagram of a TM Library

This section and a class diagram on Figure 2 describes the **basic building blocks required to design various trust management services**. We start with an *Encounter* and a *Proof*, already introduced in the previous section. An *Encounter* includes information about *Context*, about the participating *Agents*, and about the available *Actions* and their outcomes. A set of template encounter definitions can be created in the TM library when a service is developed for a specific application. Yet, the instance of an *Encounter* will be received during service invocation. In the previous section, two examples of encounters were described. In the Internet auction, an *Encounter* represents a transaction between a buyer and a seller. In the Web service, an *Encounter* is a Web service invocation or an attempt to obtain ”security tokens” from a TM authority.

Note that an Encounter can also be used to model an interaction between two agents who exchange Proofs. The treatment of exchange of *Proofs* (for example, during reporting in an auction service, during trust negotiation, or during gossiping of opinions in a P2P application) as an *Encounter* emphasizes that TM methods can be used to decide whether to trust the received *Proofs*. This form of trust is sometimes referred to as *credibility* [5], which is modeled on Figure 2 as a class that inherits from Trust. A *Proof* represents any information that can be used by diverse TM methods to compute trust. A *Proof* can store data about past encounters of the agent running the library, but also *Recommendations* or *Reports* coming from other agents. For instance, reputation systems use mainly propagated information about history of encounters, modeled as *Reports*. *Observations* refer to encounter history, as well, but are obtained first-hand by the observing agent - such as possibility exists, for example, on Wikipedia which displays history of entry modifications. On the other hand, TM methods used in Web Services use proofs that delegate trust, modeled here as *Recommendations*. It is hard to list or foresee all kinds of proofs that will be used in a TM service library; however, the kinds planned so far should be sufficient to support many divers TM methods. *Proofs* are kept in a *Proof Container* that is under a direct control of the agent running the TM service. However, when additional *Proofs* are required that are not available, the *Proof Container* can use the *Proof Discovery Protocol* to search for new proofs or request them in *Encounters* from other agents using trust negotiation. Various TM methods use different kinds of information, such as *Trust*, *Reputation*, or *Risk*. The algorithms that actually calculate this information are all modeled as *TMAlgorithms*. The arguments of *TMAlgorithms* are usually *Proofs*, but *TMAlgorithms* can also use different *Parameters* such as *User Preferences*. For example, consider *Risk*, which could depend on *Risk Aversion* as a *Parameter*.

3.1 Activities of a TM Service Library

This section and Figure 3 show **how do the classes introduced in the previous section interact with each other**.

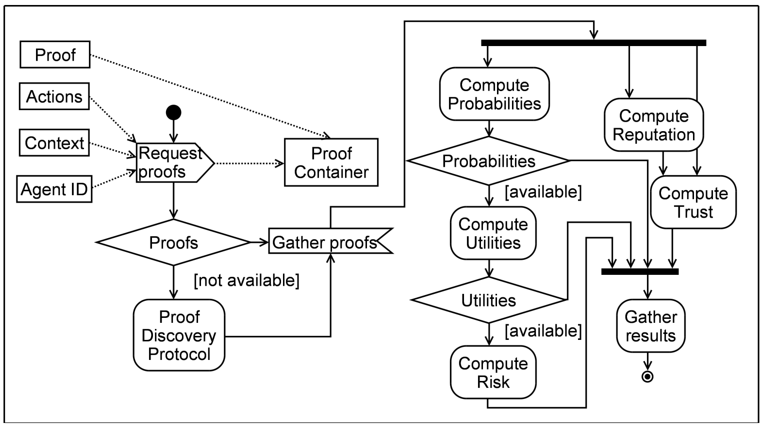

Figure 3: Activity diagram of a TM Services Library

When a TM service is invoked, it formulates a query for proofs. The query is passed to the *Proof Container* and can result in the invocation of the *Proof Dis-*

covery Protocol. Note that an *Encounter* can be a result of receiving a *Proof* from another agent; this *Proof* is forwarded to the *Proof Container* as shown on the figure. The next activity of the service library is gathering of proofs, which can then - depending on their type - be used to compute various kinds of information, using the functions stored in the library. The diagram shows the computation of Reputation, Trust and Risk, emphasizing that information about action probabilities and action utilities is necessary to calculate Risk.

3.2 Centralized or Distributed Trust Management?

In our opinion, a universal TM library should support both centralized and distributed operation. In the presented approach, *Proofs* are received during *Encounters* and stored in the *Proof Container* that is under direct control of the agent running the TM service library. This does not exclude the possibility of other agents being present who also have their own *Proof Containers*. However, agents can also use TM services provided by one, trusted agent who centrally gathers all Proofs. If a more distributed approach is used, agents can exchange Reports or issue Recommendations about other agents, using for example a P2P gossiping protocol or a structured P2P overlay, which can be instances of the Proof Discovery Protocol.

4 Evaluation of Universal TM Services

While many of the TM methods presented in literature have been evaluated by their authors, there still is no established and widely accepted method of TM evaluation - much less is there any form of a benchmark for TM methods. In the uTrust project, the development of such evaluation methods and benchmarks is planned. Here, some initial ideas are presented.

4.1 Adversary Models and TM Benchmarks

One of the most important aspects of Trust Management evaluation is the resistance of TM methods to adversaries. Since many TM methods - especially all reputation systems - are based on a majority principle, they are also vulnerable to colluding adversaries or to strategies such as discrimination [1]. Here, we discuss the characteristics of adversaries against which the TM services library should be evaluated. Adversary models depend on the **model of the environment** in which TM service works. Such a model states what basic security services are available. For example, a strong, certificate-based authentication of all agents makes many schemes of adversary behavior impossible. However, in other models of environments, only weak, pseudonymous authentication can be available. The environment model should also specify other relevant aspects of information security, for instance whether the communication primitives are vulnerable to eavesdropping, modification, or man-in-the-middle attacks. After determining an environment model, different adversary models can be chosen. An adversary can be described by the following three characteristics:

Adversary goals and maliciousness. An adversary could be selfish, pursuing only her individual gain, or she could be capable of cooperation with other adversaries. The kinds of cooperation could be limited (for example, the number of colluding agents could be limited). An adversary could also attempt to decrease the welfare of other agents (her competition) or the welfare of all agents.

Adversary knowledge. An adversary's knowledge about the TM system can vary. An omniscient adversary knows all the used TM algorithms and attempts to exploit their weaknesses; an ordinary adversary uses rather simple, sub- optimal strategies.

Adversary resources. Adversaries could have varying amounts of resources of various types: computational resources, communication resources, or even control over bots that can act as fake agents. Adversary resources would be important especially in comparison with the resources available to other agents who use the TM system.

4.2 TM Service Correctness

A TM service that consistently overvalues, or undervalues trust (or risk) certainly decreases users' performance. It is necessary, therefore, to consider TM service correctness. This could be validated by using special scenarios that can be treated analytically in order to calculate correct values of trust and risk. Next, various TM algorithms could be simulated using the considered scenario, in order to see how close the resulting values are to the correct values of trust and risk.

4.3 Evaluation of Computational Cost

A neglected, yet important aspect of Trust Management evaluation is the investigation of TM method performance. Many TM methods, in particular cryptographic and Bayesian TM, are rather complex, resulting in a non-negligible computational cost. In the context of distributed systems, some TM methods can also incur a communication overhead.

5 Conclusion

A challenge in the design of a library of universal TM services is too make it sufficiently general to express most of the TM methods known today, yet not too abstract, as it would not be possible to implement. In the service definition, more abstract interfaces allow an expression of more possible services; yet if these services have to be implemented, the interfaces must be concrete. In this paper, we have presented a blueprint of a library for universal TM services that is sufficiently general to express many of the known TM methods, yet sufficiently specific to start implementing a library of universal TM services that would also incorporate completely new TM methods. It is very hard to verify the correctness of the presented modeling approach. One method would be the development of a TM Service Library, going through an entire software development cycle. The success of the library would validate the underlying model. While we start to develop such a library and make it available as an Open Source project to the community, at present the proposed model can only be partially verified by discussions with other researchers and practitioners in Trust Management. Our works can also be thought of as a first step in the discussion of the universal Trust Management services of the future.

References

[1] Chrysanthos Dellarocas. Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In *EC ’00: Proceedings of the 2nd ACM conference on Electronic commerce*, pages 150–157, New York, NY, USA, 2000. ACM.

[2] Audun Jsang, Claudia Keser, and Theo Dimitrakos. Can we manage trust. In *Proceedings of the Third International Conference on Trust Management (iTrust), Versailes*, pages 93–107. Springer-Verlag, 2005.

[3] Lik Mui. *Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks*. PhD thesis, MIT, 2002.

[4] Sini Ruohomaa and Lea Kutvonen. Trust management survey. In Peter Herrmann, Valrie Issarny, and Simon Shiu, editors, *iTrust*, volume 3477 of *Lecture Notes in Computer Science*, pages 77–92. Springer, 2005.

[5] Sini Ruohomaa, Lea Kutvonen, and Eleni Koutrouli. Reputation management survey. In *ARES*, pages 103–111. IEEE Computer Society, 2007.