Editor: Steffen Staab
University of Karlsruhe
sst@aifb.uni-karlsruhe.de

The Pudding of Trust

Editor's Perspective

If you can spot one trend in intelligent systems, I think it would be the issue of trust. I've seen many paper titles and recently some workshops that involve the term "trust." At the same time, I've wondered what kind of trust people want their computers to manage. Intuitively, we all know that trust is important and precious, something we might work hard to earn from others and might not assign generously when things really matter. But what is trust in computing?

The literature gives numerous answers: reputation, security concerns, quality of data or services, credentials, risk management, and many more.[1] They all can play a role when dealing with trust. So, when trying to define trust in computing, we end up with a pudding of things rather than a solid definition.

To address this "pudding of trust" are this issue's five complementary contributions. Bharat Bhargava, Leszek Lilien, Arnon Rosenthal, and Marianne Winslett emphasize that rather different issues can arise with trust. When building an application, you must find the sweet spot where the trust infrastructure is neither so strong nor so weak that it would jeopardize acceptance and scalability.

Morris Sloman focuses on trust issues in pervasive applications and on the problem that arises with graded levels of trust. Tharam S. Dillon, Elizabeth Chang, and Farookh Khadeer Hussain present a concrete approach for computing such graded trust levels.

Wolfgang Nejdl and Daniel Olmedilla consider the problem of passing credentials in peer-to-peer systems. In particular, they recommend sharing credentials with third parties if trust negotiation strategies allow this.

Finally, Vipul Kashyap considers the dimension of "trusting in information," which can be a slippery issue in a world where information semantics aren't fixed but remain a moving target.

So, this issue's authors present a range of interesting thoughts and recipes on how to think about trust and deal with it. Now, as the proof of the pudding lies in the eating, enjoy!

—Steffen Staab

Reference

1. T. Grandison and M. Sloman, "A Survey of Trust in Internet Applications," *IEEE Communications Surveys and Tutorials*, vol. 3, no. 4, 2000; www.comsoc.org/livepubs/surveys/public/2000/dec/pdf/grandison.pdf.

Pervasive Trust

Bharat Bhargava and Leszek Lilien, *Purdue University*
Arnon Rosenthal, *The MITRE Corp.*
Marianne Winslett, *University of Illinois at Urbana-Champaign*

In this essay, we poke at several sacred cows (even though we like their milk and drink it ourselves) and point out several overlooked ordinary cows as well. We discuss our little herd of claims in five sections.

Using trust in computing systems is already common and should be pervasive

Trust—"reliance on the integrity, ability, or character of a person or thing"[1]—is pervasive in social systems. We constantly apply it in interactions between people, organizations, animals, and even artifacts ("Can I trust my car on this vacation trip?"). We use it instinctively and implicitly in *closed and static* systems, or consciously and explicitly in *open or dynamic* systems. An epitome for the former case is a small village, where everybody knows everybody, and the villagers instinctively use their knowledge or stereotypes to trust or distrust their neighbors. A big city exemplifies the latter case, where people use explicit rules of behavior in diverse trust relationships. A city dweller builds up trust, for instance, by asking friends or recommendation services for a dependable plumber.

If trust is so pervasive and beneficial in complex social systems, why not exploit *pervasive trust* as a paradigm in computing environments? (Using pervasive trust in nonpervasive computing is not a contradiction!) We already use trust in computing systems extensively, although usually subconsciously. Examples are users' trust-based decisions to search for reputable ISPs or e-banking sites, or to ignore emails from "Nigerians" asking for help transferring millions of dollars out of their country. The challenge for exploiting trust in computing lies in extending the use of trust-based solutions, first to artificial entities such as software agents or subsystems, then to human users' subconscious choices.

Trust is a complex, multifaceted, and context-dependent notion. Therefore, words of caution are in order. First, using the trust paradigm requires that you carefully select all and only those useful trust aspects needed for the system you're designing. Otherwise, either flexibility or performance will suffer.

Second, unwarranted demands for evidence or credentials render trust-based interactions laborious and uncomfortable, while insufficient requirements brand them too lax. (In the latter case, who wants to be friends with someone who befriends crooks and thieves?)

Third, excessive reliance on explicit trust relationships hurts performance. For example, modules in a well-integrated system should rely on implicit trust, just as villagers do. In a crowd of entities, only some communicate directly, and even fewer use trust explicitly.

The report from a National Science Foundation Information and Data Management workshop session on trust, privacy, and security raises many other issues in this research area.[2]

You can trade privacy for trust, but this requires privacy guarantees

We define *privacy* as an entity's ability to control the availability and exposure of information about itself. (Extending in this definition the subject of privacy from a person in the original definition[3] to an entity—including an organization or software—is controversial, but stimulating.) Privacy and trust are closely related. Entities can choose to trade their privacy for a corresponding gain in their partners' trust. As in social interactions, the scope of an entity's privacy disclosure should be proportional to the benefits expected from the interaction. For example, a customer applying for a mortgage must reveal much more personal data than someone buying a book.

Optimally, a party would give up only as much privacy as is indispensable for gaining the level of trust required for establishing a desirable partnership. To facilitate finding this optimum level, researchers need to propose privacy and trust measures, automate the evaluation of the privacy loss and trust gain, and quantify this trade-off.[4]

Any exchange of an entity's private information for a gain in its partners' trust depends on satisfactory limits on its further dissemination, such as a partner's solid privacy policies. Merely perceiving the potential for a partner's privacy violation makes the other entity reluctant to enter into a partnership. A user who learns that an ISP has carelessly revealed any customer's email will look for another ISP. So, the privacy-for-trust trade requires privacy guarantees.

Socially based paradigms will play a big role in pervasive-computing environments

In pervasive computing environments, people will be surrounded by zillions of computing devices of all kinds, sizes, and aptitudes.[5] Most of them will have limited or even rudimentary capabilities and will be quite small, such as radio frequency identification tags and smart dust. Most will be embedded in artifacts for everyday use, or even human bodies (with possibilities for both beneficial and apocalyptic consequences).

Unwarranted demands for evidence or credentials render trust-based interactions laborious and uncomfortable, while insufficient requirements brand them too lax.

Pervasive devices with inherent communication capabilities might even self-organize into huge, *opportunistic* sensor networks, able to spy anywhere, anytime, on everybody and everything within their midst. (The definition of the term "opportunistic"—in accordance with our intentions—suggests "often unethical" behavior.[1]) Without proper means of detection and neutralization, no one will be able to tell which and how many snoops are active, what data they collect, and who they work for (an advertiser? a nosy neighbor? Big Brother?). Questions such as "Can I trust my refrigerator?" will not be jokes—the refrigerator will be able to snitch on its owner's dietary misbehavior to the owner's doctor.

Will pervasive computing force us to abandon all hope for privacy? Will a cyberfly, with high-resolution camera eyes and supersensitive microphone ears, end privacy as we know it? Should a cyberfly be too clever to end up in the soup, the only hope might be to develop cyberspiders. But cyberbirds might eat those up. So, we'll build a cybercat. And so on and so forth …

Radically changed reality demands new approaches to computer security and privacy. Will a new privacy category appear—namely, protecting artificial entities' privacy? We believe that socially based paradigms, such as trust-based approaches, will play a big role in pervasive computing. As in social settings, solutions will vary from heavyweight ones for entities of high intelligence and capabilities (such as humans and intelligent systems) interacting in complex and important matters, to lightweight ones for less intelligent and capable entities interacting in simpler matters of lesser consequence.

Trust management has emphasized logic, not IT

Work to date focuses on providing a good target policy language for reasoning, with a few excursions into knowledge capture. These policy languages have been based on mathematical logic—typically extensions of Datalog that provide good reasoning power. But using Datalog to describe real-world security policies is like giving IT people a Lego kit—flexible, but microlevel if used to model the world.

Further complicating technology transfer, the trust management (TM) community uses various Datalog dialects, with extensions to make them better suited to knowledge capture. Logic has lost previous battles for mindshare in both the software and database arenas. Although the research community explored Prolog and Datalog, industry decided that abstraction, inheritance, and composition features were more important and went to objects. Will this happen again?

TM researchers should move beyond the question of the best prototype they can build, and be sensitive to the pragmatics of choosing a language. The list of issues is long. Is there a standard for the language, with powerful supporters? What reasoning facilities do users need, and what portion of the reasoning facilities must reside in the formal system? If you provide a strictly flat relational reasoner (for example, Datalog), can users draw conclusions about the complex objects that interest them? Systems frequently use structures more complex than relations—even relational databases are moving in that direction. Interfaces are moving to XML. How can TM work leverage these advances and produce a system that IT people and even ordinary users could use to write and maintain security-related policies?

If Tim Berners-Lee is right (and Sir Tim has a good track record), much of the knowledge that users are reasoning about will be related in OWL—for example, BloodTest will be known as a kind of MedicalTest. Perhaps TM researchers should declare a moratorium on Datalog dialects and focus on OWL. If enhancements are needed for TM purposes, should we add them to OWL?

Going beyond the formalisms, inserting TM into the wider system presents major challenges (see Rohit Khare and Adam Rifkin's *First Monday* article for a good discussion of TM on the Web[6]). How do you manage trust attributes' semantics and deal with policies that go far beyond the traditional authentication-centered public-key infrastructure and signed content? How can we extend the TM framework to encompass data quality metrics, data-cleansing techniques, and assertions of all sorts in reasoning about trust?[7] Secrecy-preserving query techniques can help but might not provide sufficient flexibility, performance, and ease of use—complete protection might be infeasible. Thus, how can we transparently insert trust and belief into query processing as an additional criterion for optimization—either as a constraint or as a quantitative trade-off? Can we provide principles for returning partial results? Across organizations, you can anticipate that terminologies, data types, operations, groups, roles, and so on will differ. Suppose someone has done the usual "semantic integration" job and produced a set of concept mappings. What does this tell us about how security policies should map across organizations?

A final major challenge is to provide the technical basis for auditing privacy-relevant behavior by corporations or government agencies. (A nontechnical basis—laws plus auditors who are trusted by both privacy advocates and the organizations whose records they will examine—will also be needed.)

Setting up a trust infrastructure (with facilities for identity management, policy evaluation, and trust establishment) is laborious but relatively easy. A policy management infrastructure is tougher, with its requirements for GUIs for composing, updating, importing, versioning, protecting, and analyzing policies. Toughest of all, and the goal of both infrastructures, is determining an appropriate organizational policy and automatically mapping that high-level policy to suitable enforcement mechanisms and to the semantic and security models that partner organizations use.

Artificial and "from scratch" application areas are for wimps

Fervor sometimes causes adherents to capitalize the name of an emerging application area to prove that it is Something Incredibly Important, as with the Semantic Web and Grid Computing. In some cases, Capitalized Research Areas eventually became real-world success stories. The World Wide Web is an excellent example. But who today runs real applications on real computational grids, and who uses a real semantic web?

To aid in establishing trust, organizations' policies describe who can do what under what circumstances, and these policies are central to establishing trust in these applications.

Hardly anyone, so far. And that, we argue, is what makes today's Capitalized Research Application Areas so appealing to researchers, including us—there aren't real users to worry about, there aren't real applications to worry about, and there are no reality checks. This lack of reality lets our creative juices flow, unchecked by legacy issues, feature interactions (including working with features designed by others), manageability requirements, precise semantics, and other pesky real-world concerns. Yet, are there enough researchers investigating the most pressing real-world security problems, such as enterprise-wide and cross-enterprise information systems?

Now that we've questioned our own research tastes, let's address the obvious follow-up question: What application areas in TM do we recommend for those who want to be considered "ironwomen" or "ironmen"? At the top of our list are *supply chain management* (SCM),[8] *customer relationship management*,[9] and *collaborative engineering* (for example, bid creation). Hundreds of vendors compete in these billion-dollar market areas, so these applications' economic importance is indisputable. Furthermore, all three involve opening up systems so that people outside a particular company can access internal corporate services and resources. For example, in an SCM system, the order entry services, order status services, and sales and production forecasts of WheelCorp could become accessible to its supplier, Bearings.com, and to WheelCorp's customer, CarsInc, as well as to Bearings.com's own suppliers and CarsInc's own customers. Market forces are driving the move to this kind of cross-enterprise integration and are pushing companies to reorganize themselves internally as *virtual enterprises*—that is, enterprises in which the means of fulfilling any particular business need can be switched in an instant between different suppliers.

As in any open system and virtual enterprise, trust is a central issue here. To aid in establishing trust, organizations' policies describe who can do what under what circumstances, and these policies are central to establishing trust in these applications. What exciting application areas these are, with their large but constantly evolving installed software bases and the potential for huge impact for successful research projects. Like other really hard applications, the issues here involve clarifying confusing situations and providing a migration path for legacy systems without designing from scratch.

Acknowledgments

Bharat Bhargava and Leszek Lilien acknowledge related research support by NSF grants IIS-0209059 and IIS-0242840, and a Cisco URP grant. Marianne Winslett acknowledges related research funding from DARPA under AFRL F33615-01-C-0336, SSCSD N660001-01-1-8908, NAI Labs F30602-97-C-0336, and NSF under CCR-0325951 and IIS-0331707.

References

1. *The American Heritage Dictionary of the English Language*, 4th ed., Houghton Mifflin, 2000.

2. B. Bhargava et al., *Trust, Privacy, and Security: Summary of a Workshop Breakout Session at the National Science Foundation Information and Data Management (IDM) Workshop held*

in Seattle, Washington, Sep. 14–16, 2003, tech. report 2003-34, Center for Education and Research in Information Assurance and Security, Purdue Univ., Dec. 2003; www.cerias.purdue.edu/tools_and_resources/bibtex_archive/archive/2003-34.pdf.

3. "Internet Security Glossary," *The Internet Society*, Aug. 2004; www.faqs.org/rfcs/rfc2828.html.
4. B. Bhargava and L. Lilien "Private and Trusted Collaborations," to appear in *Secure Knowledge Management* (SKM 2004)*: A Workshop*, 2004.
5. "Sensor Nation: Special Report," *IEEE Spectrum*, vol. 41, no. 7, 2004.
6. R. Khare and A. Rifkin, "Trust Management on the World Wide Web," *First Monday*, vol. 3, no. 6, 1998; www.firstmonday.dk/issues/issue3_6/khare.
7. M. Richardson, R. Agrawal, and P. Domingos, "Trust Management for the Semantic Web," *Proc. 2nd Int'l Semantic Web Conf.*, LNCS 2870, Springer-Verlag, 2003, pp. 351–368.
8. P. Schiegg et al., "Supply Chain Management Systems—A Survey of the State of the Art," *Collaborative Systems for Production Management: Proc. 8th Int'l Conf. Advances in Production Management Systems* (APMS 2002), IFIP Conf. Proc. 257, Kluwer, 2002.
9. N.C. Romano Jr. and J. Fjermestad, "Electronic Commerce Customer Relationship Management: A Research Agenda," *Information Technology and Management*, vol. 4, nos. 2–3, 2003, pp. 233–258.

Trust Management in Internet and Pervasive Systems

Morris Sloman, *Imperial College London*

Businesses, professionals, and scientists will increasingly use the Internet to set up virtual organizations to provide a service or support collaboration. Examples include combining the distributed databases and processing services owned by different organizations for an e-science application, and the organizations that collaborate on designing and manufacturing components for automobiles or aerospace. For existing collaborations, trust is based on many years' experience, but setting up new collaborations requires evaluating trust without the benefit of face-to-face meetings, which provide considerable but subtle information that lets people assess mutual trust.

In future pervasive-computing applications, mobile devices will interact with each other and with an intelligent infrastructure to help people with normal activities. An example is ubiquitous healthcare, which will be able to monitor an individual's chronic illness, such as a heart condition, diabetes, or epilepsy; determine abnormal events; and notify the patient as well as medical or emergency services as necessary. We'll use the intelligent environment for sensing context and providing additional processing or storage services, as well as for remote communication. Pervasive applications might require dynamic ad hoc collaborations as people move around and join or leave them. Obvious implications exist when it comes to trusting the intelligent environment, particularly with respect to potentially sensitive medical data.

In future pervasive-computing applications, mobile devices will interact with each other and with an intelligent infrastructure to help people with normal activities.

From these scenarios, you can see that systems must be able to assess trust and use this as the basis for automated decision making—whether to use a service, whether to permit access to resources in an ad hoc collaboration or virtual organization, and what type of security mechanisms (for example, encryption or authentication) a person should use when interacting with unknown services or collaborators.

Trust components

We define trust as "the quantified belief by a trustor with respect to the competence, honesty, security, and dependability of a trustee within a specified context."[1] Quantification reflects that a trustor can have various degrees of trust, which could be expressed as a numerical range or as a simple semantic classification. Specifying actual trust values can be difficult, so we often use classes such as high, medium, and low. We can use a numerical range to reflect uncertainty in the assessment of the trust value—for example, we might assign a large range initially but, owing to the experience of interacting with the trustee, we could reduce the range to reflect increased certainty (confidence) in the trust value.[2]

Trust is specified in a specific context.[3] For example, a trustee might use resources such as data the trustor owns, or provide a specific service to the trustor such as medical advice, processing, or authentication. A trust level for one context doesn't normally apply to a different context, so you wouldn't trust a highly trusted medical advisor for financial advice. The attributes of trust depend on the trust context. Honesty and truthfulness might be more important for financial trust relationships, but competence would be a key attribute for a medical advisor. Dependability would encompass reliability and timeliness, which could be important for real-time applications. Security would be a key attribute for a trusted medical monitoring service.

Distrust might be a useful concept to indicate entities that you should avoid for interaction purposes. A simple way to represent distrust is with negative trust values.

Generally, trust isn't transitive. If B trusts C, and A trusts B, this doesn't imply that A trusts C. However, A could delegate trust decisions to B. For example, B could be an authentication server or certification authority. A will trust an entity that B trusts and certifies, although A might not know the basis of B's trust. Another form of trust transitivity occurs when A uses a service B that depends on another service C. A implicitly trusts C but might not know that B depends on C, and so might not know of this implicit trust relationship.

When first encountering an unknown entity, a trustor can rely on recommendations from other trusted entities. Recommendation (used in electronic auction and peer-to-peer systems[4]) differs from delegation in that the delegatee actually makes the trust decision for the trustor, but a trustor uses recommendations as additional information to make its own trust decision. Reputation is usually based on a combination of recommendations and experience from many different third-party sources. Various techniques exist for combining recommendations: using simple averages, letting old recommendations decay, weighting them on the basis of the recommender's trust levels, and so on.

Trust isn't a static concept. It changes over time owing to experience from the outcome of interactions between the trustor and trustee, or as a result of other entities' experience interacting with the trustee and modifying its reputation.

Trust is also related to risk,[5] where risk is the probability of loss with respect to an interaction—for example, nonpayment for a service or a server failing and losing information. We might use risk related to a context to determine the level of trust, but we must also consider the interaction's value. I might be prepared to trust a high-risk supplier for a low-valued \$10 CD purchase, but I wouldn't trust a medium-risk supplier for a high-valued \$20,000 car purchase. Some systems, such as finance decision making, use the risk value as the basis for decision making. In this case, a trust value might be an element in the risk valuation.

Trust management is collecting, codifying, analyzing, and evaluating evidence relating to competence, honesty, security, or dependability with the purpose of making assessments and decisions regarding trust relationships. Trust management systems must support analysis of trust and recommendation specifications to detect conflicts and inconsistencies and support trust queries related to decision making.[1]

Trust and recommendation specifications

We need a precise specification of trust and recommendation requirements so that automated components can use this information to make decisions.[1] Thus,

trust (*Tr*, *Te*, *As*, *L*) ← *Cs*

states that a trustor *Tr* trusts or distrusts a trustee *Te* to perform actions *As* at trust range *L* if constraint *Cs* is true. *As*, the action set, is a colon-delimited list of actions (which effectively specify the context). An action, such as storing information, could be performed by the trustor on the trustee, or by the trustee on the trustor. Actions might have parameters (not shown earlier), the first of which indicates whether the action is performed on the trustor or trustee. *L*, the range of trust levels, is expressed as integers between 0 and 100. Negative values represent distrust. We can specify *L* as predefined labels—for example, high trust is 90 to 100, low trust is 5 to 20, and a default initial trust range is 0 to 50. When the trust management system evaluates a trust relationship, it only applies if the set of constraints *Cs* evaluates to true. Elements in *Cs* can be combined by logical And (&) and logical Or (|). For example, in

```
trust(AnalProg, StorageServer,
StoreData(StorageServer), 80–100 )
← StorageServer.owner = CoXYZ &
risk(StorageServerFail) < 0.1
```

the analysis program (AnalProg) trusts the storage server for storing data when the storage server is owned by CoXYZ and risk of the server failing is less than 0.1. (We assume risk has a value between 0 and 1.) The recommendation

recommend (*Rr*, *Re*, *As*, *L*) ← *Cs*

Trust management systems must support analysis of trust and recommendation specifications to detect conflicts and inconsistencies and support trust queries related to decision making.

states that the recommender *Rr* recommends the recommendee *Re* at recommendation level *L* to perform *As* if constraint *Cs* is true. *As*, the recommended action set, is a colon-delimited set of actions defining a context for the recommendation. *L* can be a range of values and *Cs* a delimited set of constraints, as in the trust rule. Negative values for recommendations indicate that the recommender doesn't recommend the recommendee.

Note that we assume that the recommendation and trust levels are independent of each other. A high-level recommendation could result in a low trust level. A trust rule constraint could be based on a recommendation, or a recommendation constraint could be based on a trust rule. In

```
recommend (AmbulanceSupervisor,
_Paramedic, JoinResponseTeam, high)
← employed(LondonAmbulance, _Paramedic)
```

the AmbulanceSupervisor recommends letting any paramedic that LondonAmbulance employs join the emergency response team at an accident.

Trust-based decisions

You can use the trust specification to influence decisions in many ways. When forming virtual organizations and ad hoc dynamic communities, we must establish trust between the various entities that will constitute the virtual organization before negotiating service level agreements or contracts. A clear requirements specification for this trust is needed in terms of trust rules. This could be based on recommendations, assertions, or signed credentials from third parties.[6]

You could refine trust specifications into security policies. For example, if the trust rules indicate that the network infrastructure isn't secure, you'll need a strong form of encryption for all external communication, and you must use mutual authentication. An authorization policy can also explicitly make access control decisions on the basis of querying a trust level from a trust management system. A Ponder authorization policy[7] specifies the actions that a subject can perform on a target object when an optional constraint is true. The following Ponder authorization policy specifies that only clients with high trust levels can access a prerelease section of the music content base. Access is denied if the policy constraint, which queries the trust management system to evaluate the trust level, determines it isn't high.

```
Inst auth+ Access {
   subject Client;
   target ContentBase/PreRelease;
   action AccessMusic();
   when (trust (Client, AccessMusic)
      >=high }.
```

You could base recommendations on current trust ratings for an entity either in the same context or with respect to a different context—that is, no trust rules exist for that specific context. For example, Fred gets a request for a recommendation of eTunes as a video content supplier but only has a trust rule related to music supply. Fred isn't sure about eTune's ability to provide the data rate needed for video, so he only gives a low rating for a recommendation.

In ad hoc collaborations between mobile

entities, some of the members might not have Internet access. Thus, it isn't always practical for an existing member to evaluate a new member's credentials by accessing the credential-issuing authority to obtain its public keys. In such a situation, it might be necessary to trust other members of the collaboration to help with the validation. For example, other members might actually have the public key required to validate the credential or might be able to recommend the new entity on the basis of some other information.[6]

Trust issues

Many unsolved trust issues exist in Internet and pervasive applications.

A big problem in automating trust-based decisions is evaluating a trust level or range. Determining initial trust values can be quite difficult, and the default values might be rather arbitrary. Combining evidence related to experience, recommendations, reputation, and risk into a trust evaluation can be complex and application dependent. In many practical situations, there might not be any past experience for a specific trustee or context on which to base a trust evaluation. So, the evaluation might have to depend on trust evaluation from a different context. How to do this isn't obvious. A possibility exists for collusion between agents to provide false recommendations, so a need exists for trust evaluation of recommenders,[4] which can lead to circular dependencies between trust and recommendations.

The privacy issue is a fundamental problem in pervasive systems, which inherently track information such as activity, location, and various other personal information. In most cases, the pervasive infrastructure is responsible for this tracking (as in cellular phone systems). How can you trust the organizations managing the infrastructure to use this context information responsibly and not pass it on to inappropriate third parties?[8] The same issue applies to any organization trusted to monitor personal or medical information.

Using anonymity or pseudonymity to support privacy in pervasive systems has trust implications as well. We must support trust specifications and reasoning for anonymous entities, but this can cause problems with respect to payment for services because many payment systems depend on identities.

In pervasive systems, users move around and might be at home, at the office, in a public building, or on the street. The level of trust in the infrastructure or related to personal-area networks belonging to people in the vicinity will depend on the current context—where you are, what you're doing, who's in the vicinity, and so on. Trust is thus context dependent and should be considered when adapting security policies to the current context. For example, when in a meeting, I might trust the attendees to access services in my personal area network, but when walking on the street, passersby shouldn't have any access.

Setting up a virtual organization for e-commerce applications or to provide a consolidated service from multiple service suppliers requires negotiating contracts based on trust. This could involve interactions between composed or federated trust domains. The contracts will form a basis for policies relating to the entities' rights and duties within the virtual organization and service level agreements for customers. This must map to policies governing the interactions, which must also adapt as trust changes with experience, risk, or transaction value. Automated techniques and tools to support this don't currently exist.

We must support trust specifications and reasoning for anonymous entities, but this can cause problems with respect to payment for services because many payment systems depend on identities.

Acknowledgments

I gratefully acknowledge support from the European Union for iTrust and Trustcom projects.

References

1. T. Grandison and M. Sloman, "Trust Management Tools for Internet Applications," *Proc. 1st Int'l Conf. Trust Management*, LNCS 2692, Springer-Verlag, 2003, pp. 91–107.
2. C. English, S. Terzis, and W. Wagealla, "Engineering Trust Based Collaborations in a Global Computing Environment," *Proc. 2nd Int'l Conf. Trust Management,* LNCS, Springer-Verlag, 2004, pp. 120–134.
3. T. Grandison and M. Sloman, "A Survey of Trust in Internet Applications," *IEEE Communications Surveys and Tutorials*, vol. 3, no. 4, 2000; www.comsoc.org/livepubs/surveys/public/2000/dec/index.html.
4. L. Xiong and L. Liu, "PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities," *IEEE Trans. Knowledge and Data Eng.*, vol. 16, no. 7, 2004, pp. 843–857.
5. A. Josang and S. LoPresti, "Analysing the Relationship between Risk and Trust," *Proc. 2nd Int'l Conf. Trust Management*, LNCS, Springer-Verlag, 2004, pp. 135–145.
6. S. Keoh and E. Lupu, "PEACE: A Policy-Based Establishment of Ad-Hoc Communities," *Ann. Computer Security Conf.* (ACSAC 04), 2004.
7. N. Damianou et al., "The Ponder Specification Language," *Proc. Policy 2001: Workshop Policies for Distributed Systems and Networks*, LNCS 1995, Springer-Verlag, 2001, pp. 18–39.
8. J. Seigneur and C. Jensen, "Trading Privacy for Trust," *Proc. 2nd Int'l Conf. Trust Management*, LNCS, Springer-Verlag, 2004, pp. 93–107.

Managing the Dynamic Nature of Trust

Tharam S. Dillon, *University of Technology, Sydney*
Elizabeth Chang and Farookh Khadeer Hussain, *Curtin University of Technology, Australia*

Trust has been an important element in interpersonal relationships in many fields. In computing, it's only become really important with the Internet's widespread use. A significant difference in the virtual world is the absence of physical cues that form the basis of trust in the physical world. Stephen Marsh defined trust early for the distributed artificial intelligence field.[1] Recently, trust issues have taken on more urgency owing to four factors:

- Peer-to-peer (P2P) communication, possibly anonymous
- Virtual communities, which must protect community integrity[2]

Table 1. The factors that define the dynamism of trust.

Factors	Nature	Specifiable, observable, and measurable?
Actual behavior, expected behavior	Peers' external factors	Yes
Willingness, capability	Peers' internal factors	No
Context, time, association type,	Relationship factors initiation relationship	Yes

- E-commerce
- Cyberterrorism, which aims to disrupt services

Trust is sometimes used synonymously with security, but security and trust are two distinct concepts. Security in the virtual world usually refers to enabling sheltered communication between two entities. On the other hand, we can imagine situations where two interacting parties can resort to unfair practices over a sheltered, secure communication setup. Trust helps determine the likelihood of detecting and preventing such practices. As we're all aware, trust between two real-world parties tends to vary with time, giving it a dynamic character. Similarly, trust in the virtual world also has a dynamic aspect, and this essay examines the management of these dynamic aspects of trust.

Trust, trustworthiness, and trust ontology

When examining trust, we're frequently concerned with P2P systems because of individual peers' autonomy and the decentralized nature of P2P communication. Additionally, P2P communication might be either anonymous or psuedononymous or nonanonymous.[3] Two important and related concepts in such systems are trust and trustworthiness.[4]

We define trust as the trusting peer's belief in the trusted peer's willingness and capability to behave as expected by the trusting peer in a given context at a given time slot in a particular association relationship.[3,4] The notions of *belief*, *context*, *time slot*, *willingness* and *capability*, and *as expected by the trusting peer* are very important to this definition.

To characterize trust, we must have some way to measure it.[4] We define trustworthiness as a measure that depicts the level of trust that the trusting peer has in the trusted peer in a given context at a given time slot with a given type of association relationship. Noting this, we can define a *trust ontology* as

Trust [trusting peer, trusted peer, context, time slot start time, time slot end time, association type, trustworthiness value]

The association relationship distinguishes whether the trustworthiness measure is between two individual peers, a peer and a group of peers, two groups of peers, or within a group of peers. Trustworthiness itself is determined through

- Direct interaction between the trusting and trusted peers using CCCI metrics[4]
- Use of the trusted peer's reputation as based on the opinion of *witness peers* that have previously interacted with the trusted peer
- Historical records of previous interactions with the trusted peer

Reputation is widely used when the trusting peer has had no previous interaction with the trusted peer, just as we find in the real world.

The dynamic nature of trust

As we noted earlier, as time passes, the trust one entity has in another might not stay the same and could change owing to the following factors:

- After further dealings, the trusting entity has a better idea of the trusted entity's capability and willingness to act the way the trusting entity wants in a given context.
- The trusted entity's capability or willingness to act in a given context the way the trusting entity desires might change with time.
- The trusting entity, after getting recommendations from other entities, will know more about the trusted entity's capability and willingness to act the way that the trusting entity wants in a given context.

We define the *dynamic nature of trust* as the change in the trustworthiness value of an entity, assigned to it by a given trusting entity with the passage of time in different time slots.

Because capability and willingness are by and large not directly observable, we must estimate them using peers' external factors from within the relationship characteristics (see Table 1).

Managing trust dynamics

As we noted, reputation is one of the most widely used mechanisms for computing trustworthiness.[5–8] However, given the dynamic nature of trust with peers, we must find a mechanism for predicting reputation value in future time slots.

First, let's define some terms necessary for discussion. The *repute value* denotes an entity's reputation in a given context and at a given time spot as communicated by a witness entity. *Trustworthiness prediction* is the process of determining the trusted entity's future trustworthiness value given its past repute values. A trustworthiness prediction's *time space* is the total duration of time over which the trusting entity will analyze the trusted entity's behavior and process the trustworthiness prediction. The *time spot* is the time at which an entity interacted with another entity and subsequently assigned it a trustworthiness value. A trustworthiness prediction's *time slot* is the breadth of time over which the reputation-queried entity's repute values are aggregated into a single value for analyzing its dynamic behavior.

An entity will have a repute value for each time slot, and the trusting entity will use these values to predict the entity's future trustworthiness value.

We define the *witness trustworthiness value* as a numeric value that denotes the correctness of the witness entity's recommendations. Numerically, it's the average of the difference between the value the witness entity communicates about the reputation-queried entity and the trustworthiness value that the reputation-querying entity found by interacting with the reputation-queried entity.

We use a trustworthiness scale [0, 6], in which each numerical value denotes a level of trust.[4] We use the range [1, 2] to denote negative trust, [3, 4] to denote neutral trust, and [5, 6] to denote positive trust. We assign the value of 0 to newcomers. When a new entity joins the entity network and carries out transactions with other entities, irrespective of whether its behavior in those transactions was good or bad, the entity gets a trustworthiness value greater than 0.[4] So, it has little motivation to drop the identity with which its reputation is associated and come back as a newcomer with a new identity.

The range of witness trustworthiness value in our proposed method, therefore, is [–5, 5]. We assume that a witness entity with a repute value in the [–1, 1] range is communicating accurate recommendations. A repute value in the [–1, 1] range occurs because the difference between the two factors that determine the witness trustworthiness value of an entity is at most one level of trustworthiness.

We consider three scenarios in which the trusting entity must decide whether to interact with another entity.

Case 1

The trusting entity must make a trust decision within the current time slot N and knows the trusted entity's trustworthiness value in that time slot.

The trusting entity determines if it previously interacted with the trusted entity. If it has, and if the time spot of previous interaction and the time at which it must make its trust-based decision fall in the same time slot (the last time slot N), then it doesn't need to issue a reputation query for the trusted entity. It uses the value that it holds to make a trust-based decision.

Case 2

The trusting entity must make a trust decision within the current time slot N and doesn't know about the trusted entity's trustworthiness value in that time slot.

In this case, the trusting entity must issue a reputation query for the prospective trusted entity and specify the context in which it wants to interact with the other entity, along with the time space.

The reputation-querying entity first classifies the obtained reputation into the different time slots using the time spot when the interaction took place. It uses the witness entities' trustworthiness values to weed out reputation from the untrusted entities. It then combines the reputation obtained from trustworthy and unknown entities using the following expression to determine the reputation-queried entity's trustworthiness at each time slot:

$$\text{Repute value}\,(m, A, T_A) = \frac{\left(\sum_{i=1}^{N} (WTV[i]) \Diamond \left(\text{Repute value}[m, i, A, t]\right)\right)}{N} + \beta * \left(\sum_{j=1}^{M} \text{Trustworthiness}[j]\right) / M, \qquad (1)$$

where N and M are the number of trusted and number of unknown entities, respectively, in time slot T_A, A denotes the context, T_A denotes the time slot on the time space, T denotes the time spot that the reputation-querying peer finds in time slot T_A, and m denotes the identity of the reputation-queried peer. Repute value [m, i, A, t] is the trustworthiness value that witness entity i communicates about the reputation-queried entity m at time spot t, in context A. $WTV[i]$ represents the witness i's trustworthiness value as perceived by the reputation-querying peer in the context of communicating recommendations. $\Diamond$ is an operator that adjusts the trustworthiness value communicated by the witness entity with the witness entity's witness trustworthiness value. β gives an appropriate weighting to the recommendations that the unknown entities communicated.

The first term in Equation 1 combines the reputation-queried entity's reputation values with the trusted entity's reputation values (the reputation-querying entity trusts the trusted entity to communicate accurate recommendations). The second term combines the reputation values from the unknown peers (with whom the reputation-querying entity has no previous experience soliciting recommendations).

We can define a trust ontology as Trust [trusting peer, trusted peer, context, time slot start time, time slot end time, association type, trustworthiness value]

Case 3

The trusting entity has to make a trustworthiness prediction for the trustworthiness value at the time slot $N + 1$. In this case, the trusting entity must model the dynamic nature of trust while predicting the trustworthiness value at the time slot $N + 1$.

If the trust-based decision falls at a time in the future, then the trusting entity must use the Markov method. It adopts the procedure we just explained to gather the reputation-queried entity's repute values and subsequently weeds them out for each time slot from 0 to N.

We define a reputation Markov chain as a given entity's sequence of aggregated repute values that correspond to a sequence of time slots. We can use Equation 1 to obtain the Markov chain for an entity for a sequence of time slots from 0 to N.

Then, the trusting entity must construct the current state vector **c** and the Markov matrix $\boldsymbol{M}$ to make a trustworthiness prediction for the next time slot $N + 1$ using the Markov model.

***Constructing current state vector* c.** The current state vector shows the reputation-queried entity's repute value at time slot N. It will be a 1×6 matrix because we use six trustworthiness levels. We determine the reputation queried entity's repute value at time slot N using Equation 1, denoting it with a 1 in the column corresponding to the repute value at time slot N. We denote all other trustworthiness levels with 0.

***Constructing the Markov matrix*.** A given entity's Markov matrix denotes the probability of its transiting from one trustworthiness level to another on the basis of its past behavior, which we capture using the Markov chain. To determine the probability of an entity transiting from trustworthiness Level 1 to trustworthiness Level 2, we find the ratio between the number of times that entity has transited from Level 1 to Level 2 and the total number of times the entity has transited from Level 1 to any other level. We denote an entity's Markov matrix as $\boldsymbol{M}$. This would be a 6×6 matrix with rows corresponding to the trustworthiness level at time slot N and columns corresponding to the trustworthiness level at time slot $N + 1$ of a given entity. An element in the matrix denotes the probability of the entity transiting from the trustworthiness level corresponding to the row in which the element occurs at time slot N to the column in which the element occurs, at time slot $N + 1$.

Determining the future trustworthiness value at time slot N + 1. Once we determine the Markov matrix and the current state, we determine the entity's future trust state vector by multiplying the current state vector with the Markov matrix. The future state vector denotes the probability that the entity will behave with a trustworthiness level I at time slot $N + 1$. We denote an entity's future state vector as **f**. Mathematically, we represent this as

$$\mathbf{f} = \mathbf{c} \times \boldsymbol{M}. \tag{2}$$

From the future state vector, we choose the reputation-queried entity's future trustworthiness level (at time slot $N + 1$) as the level to which the entity has the highest probability of transiting. A trusting entity decides to go ahead and interact with the trusted entity only if the trusted entity's trustworthiness level is ≥ 5 because 5 and 6 denote positive trust. This is the same for all three cases we've described.

After interacting with the chosen entity, the trusting entity uses CCCI metrics[4] to rate the trusted entity's behavior in the interaction. On the basis of the trustworthiness value assigned to the trusted entity after interacting, it, the trusting entity modifies all the witness entities' witness trustworthiness values, from which it had solicited recommendations using the following:

$$\begin{aligned}&\text{Witness trustworthiness value } (i)\\&\leftarrow \Theta \times \left(\sum_{i=1}^{z} \left(X_i - U_i \right) / Z \right)\\&+ \beta \times \left(\sum_{j=1}^{c} \left(X_j - U_j \right) / C \right),\end{aligned} \tag{3}$$

where witness trustworthiness value (i) is the trustworthiness of witness entity i in the context of communicating recommendations, X_i is the trustworthiness value that witness entity i communicates about the prospective trusted entity, U_i is the trust value that the trusting entity found when it interacted with the trusted entity, Θ and β are weights that give more importance to recent experience that an entity has with a given witness entity in soliciting recommendations than to old experiences. In general, $\Theta >> \beta$ and $\Theta + \beta = 1$. The weights Θ and β ensure that old reputation matches receive less importance or no importance ($\beta = 0$) in determining a witness entity's repute value.

Discussion and experimental results

We carried out experiments to validate our approach using the Jena multiagent system. We took 128 agents for simulation and bootstrapped the system to the same level as previous researchers.[6–10] The population of malicious agents was 10 percent of the total agent population. We took four agents whose trustworthiness values were Level 6 as trusting entities. We prompted a trusting entity chosen randomly from the four agents to carry out a transaction. Our simulation results show that 100 percent of the transactions that the trusting entities carried out were correct in this experiment. This represents an improvement over previously reported results.[6–10]

Our simulation results show that 100 percent of the transactions that the trusting entities carried out were correct in this experiment.

References

1. S. Marsh, *Formalizing Trust as a Computational Concept*, doctoral dissertation, Dept. of Computing Science, Univ. of Sterling, 1994.
2. A. Abdul-Rahman and S. Hailes, "Supporting Trust in Virtual Communities," *Proc. 33rd Hawaii Int'l Conf. System Sciences* (HICSS), IEEE CS Press, 2000.
3. F. Hussain, E. Chang, and T.S. Dillon, "Classification of Trust in Peer-to-Peer (P2P) Communication," *Int'l J. Computer Systems Science and Eng.*, vol. 19, no. 2, 2004, pp. 59–72.
4. F. Hussain, E. Chang, and T.S. Dillon, "Trustworthiness and CCCI Metrics for Assigning Trustworthiness in P2P Communication," *Int'l J. Computer Systems Science and Eng.*, vol. 19, no 4, 2004, pp. 95–112.
5. B. Yu and M.P. Singh, "Distributed Reputation Management for Electronic Commerce," *Computational Intelligence*, vol. 18, no. 4, 2002, pp. 535–549.
6. S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina, "The EigenTrust Algorithm for Reputation Management in P2P Networks," *Proc. 12th Int'l World Wide Web Conf.*, ACM Press, 2003.
7. Y. Wang and J. Vassileva, "Bayesian Network-Based Trust Model in Peer-to-Peer Networks," *Proc. Workshop Deception, Fraud, and Trust in Agent Societies, Autonomous Agents and Multi-Agent Systems Conf.* (AAMAS 03), 2003.
8. E. Damiani et al., "Managing and Sharing Servents' Reputations in P2P Systems," *IEEE Trans. Knowledge and Data Eng.*, vol. 15, no. 4, 2003, pp. 840–854.
9. L. Xiong and L. Liu, "PeerTrust: Supporting Reputation-Based Trust in Peer-to-Peer Communities," to appear in *IEEE Trans. Knowledge and Data Eng.*, special issue on peer-to-peer based data management, 2004.
10. K. Aberer and Z. Despotovic, "Managing Trust in a Peer-to-Peer Information System," *Proc. Conf. Information and Knowledge Management* (CIKM 2001), 2001, pp. 310–317.

Protecting Sensitive Data in Peer-to-Peer Networks

Wolfgang Nejdl and Daniel Olmedilla, *L3S Research Center and University of Hanover, Germany*

Peer-to-peer data and services sharing has become popular in recent years. Approaches such as Edutella[1] and Piazza[2] have started to extend these ideas toward decentralized infrastructures for general information-sharing purposes, which are most suitable for sharing data and services on the Semantic Web. These P2P networks' most important and intriguing characteristic is their dynamic nature: peers can join and leave the network as they like, and the underlying infrastructure is responsible for making peers' data and services available as long as they're part of the network.

While one interesting question is "How do we efficiently find data in a P2P network?" another is "How do we protect data in a P2P network?" This question becomes relevant as soon as the P2P infrastructure manages not only freely available data but also data meant to be accessible to only some users. Consider the following scenario, which we've investigated in the context of the EU/IST-funded ELENA project (www.elena-project.org). ELENA aims to provide personalized and decentralized infrastructures and brokerage platforms for e-learning.

Suppose that E-Learn Associates manages a Spanish course in the P2P-based ELENA network, and Alice wishes to take the course. E-Learn doesn't offer its courses for free, so Alice must use her credit card to charge the price of the course. In addition, E-Learn has an agreement with Hanover University and offers a discount to all the employees working there. Alice works at Hanover University, and she doesn't mind showing her employee ID to anyone. However, she doesn't feel comfortable showing her credit card to just anyone—she is only willing to show her credit card to companies that belong to the Better Business Bureau Online Organization.

To handle this problem, we need digital credentials, access control policies, and the ability to express *trust negotiation* between peers.[3] To see an appropriate subset of Alice's digital credentials, E-Learn must show that it satisfies the access control policies for each of them. In the process of demonstrating that it satisfies those policies, it might have to disclose additional credentials of its own, but only after Alice demonstrates that she satisfies the access control policies for each of those additional credentials, and so on, resulting in an iterative trust negotiation process between Alice and E-Learn.

```
E-Learn:
      accessCourse(Course_Name) $ Requester ←
            price(Course_Name, Price),
            appliedDiscount(Course_Name, Requester, Price, FinalPrice),
            creditCard(Requester, Number, ExpDate) @ Issuer @ Requester,
            acceptedCreditCardIssuer(Issuer),
            amountAvailable(FinalPrice, Requester, Number, ExpDate) @ Issuer.
      appliedDiscount("Spanish Course", Requester, Price, FinalPrice) ←
            employeeId(Requester) @ "Hanover University" @ Requester,
            FinalPrice = 0.8 * Price.
      appliedDiscount(Course, Requester, Price, Price).
      acceptedCreditCardIssuer("VISA").
      acceptedCreditCardIssuer("MasterCard").
      acceptedCreditCardIssuer("AmEx").
      member("E-Learn") @ "Better Business Bureau"
            signedBy["Better Business Bureau"].

Alice:
      employeeId("Alice Smith") @ "Hanover University"
            signedBy ["Hanover University"].
      creditCard("Alice Smith", 12345678, "02/05") @ "VISA" $ Requester ←
            member(Requester) @ "Better Business Bureau" @ Requester.
```

Figure 1. The PeerTrust policies governing the negotiation between E-Learn and Alice.

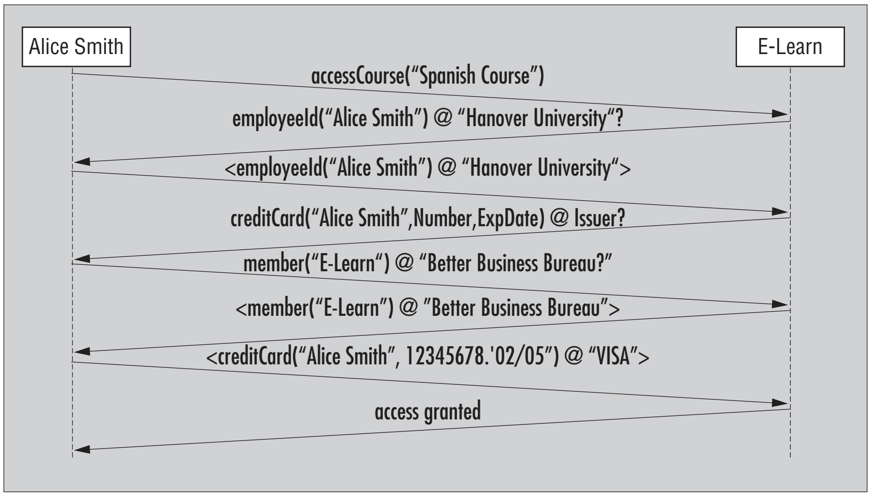

Alice Smith
E-Learn
accessCourse("Spanish Course")
employeeId("Alice Smith") @ "Hanover University"?
<employeeId("Alice Smith") @ "Hanover University">
creditCard("Alice Smith",Number,ExpDate) @ Issuer?
member("E-Learn") @ "Better Business Bureau?"
<member("E-Learn") @ "Better Business Bureau">
<creditCard("Alice Smith", 12345678.'02/05") @ "VISA">
access granted

Figure 2. Negotiation between Alice and E-Learn.

The PeerTrust project

The PeerTrust project[4,5] is investigating trust negotiation in Semantic Web and P2P environments. Within the program, digital credentials can be signed XML or RDF statements that express peer properties, and policies are expressed as logic programs that tie resource access to required credentials. The ability to refer to peers, to credentials, or to other resources in PeerTrust logic programs lets us express the iterative exchange of credentials during a trust negotiation process.

For our example, the PeerTrust policies governing the negotiation between E-Learn and Alice appear in Figure 1. An example of the interaction diagram between them is shown in Figure 2. Readers interested in experimenting with PeerTrust should see the sidebar "The PeerTrust Prototype."

As Figure 3 shows, Alice and E-Learn obtain trust negotiation software signed by a source that they trust (PeerTrust Inc.) and distributed either as a Java application or an applet. After Alice requests the Spanish course from E-Learn's Web front end, she enters into a trust negotiation with E-Learn's negotiation server. The negotiation servers can also act as servers for the resources they protect (the learning management servers), or they can be separate entities, as in our figure. Additional parties can participate in the negotiation, if necessary (the Institution A and Institution B servers in Figure 3). If Alice is granted access to the course, E-Learn sets up a temporary account for her at the course provider's site and transparently redirects her original request there. The temporary account is invisible to Alice.

Logic programs as policy languages

Logic programs are an obvious substrate

The PeerTrust Prototype

The PeerTrust 1.0 prototype is available free at www.l3s.de/peertrust or https://sourceforge.net/projects/peertrust. PeerTrust 1.0's outer layer is a signed Java application or applet program. It keeps queues of propositions that are in the process of being proved, parses incoming queries, translates them to the PeerTrust language, and passes them to the inner layer. Its inner layer answers queries by reasoning about PeerTrust policy rules and certificates using Prolog metainterpreters (in MINERVA Prolog, whose Java implementation offers excellent portability) and returns the answers to the outer layer. PeerTrust 1.0 imports RDF metadata to represent policies for access to resources and uses X.509 certificates and the Java Cryptography Architecture for signatures. It employs secure socket connections between negotiating parties, and its facilities for communication and access to security-related libraries are in Java.

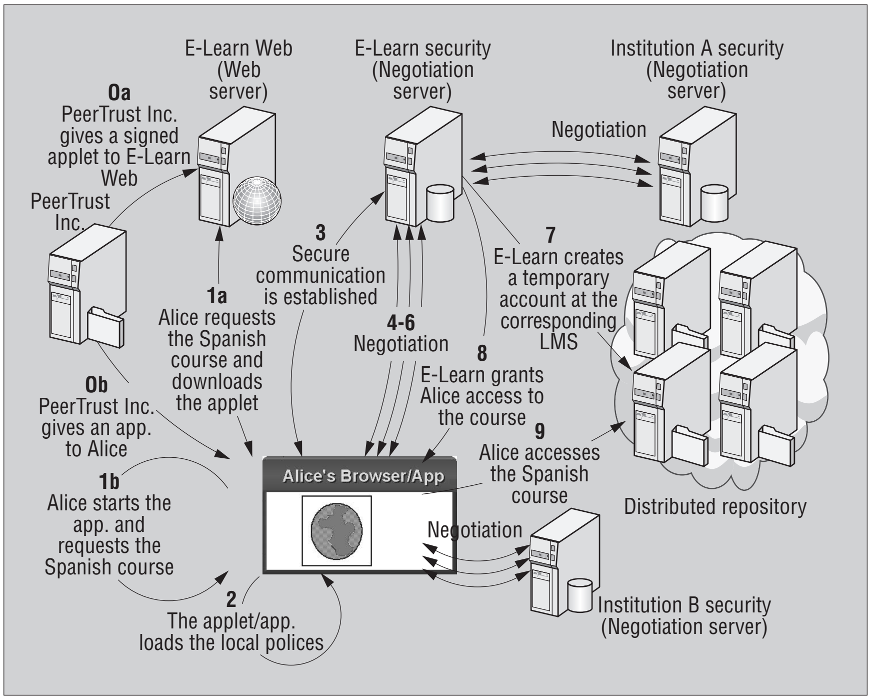

E-Learn Web (Web server)
E-Learn security (Negotiation server)
Institution A security (Negotiation server)
0a PeerTrust Inc. gives a signed applet to E-Learn Web
PeerTrust Inc.
Negotiation
3 Secure communication is established
7 E-Learn creates a temporary account at the corresponding LMS
1a Alice requests the Spanish course and downloads the applet
4-6 Negotiation
8 E-Learn grants Alice access to the course
0b PeerTrust Inc. gives an app. to Alice
9 Alice accesses the Spanish course
1b Alice starts the app. and requests the Spanish course
Alice's Browser/App
Distributed repository
Negotiation
2 The applet/app. loads the local polices
Institution B security (Negotiation server)

Figure 3. Resource access control scenario.

for declarative, universally comprehensible policy languages. In fact, recent efforts to develop policy languages suitable for use in trust negotiation have all chosen logic programming as a substrate: SD3,[6] RT,[7] Cassandra,[8] and Piero Bonatti and Pierangela Samarati's approach[9] demonstrate an emerging consensus that constraint logic programs are a promising choice.

SD3 is a trust-management system that uses an extension of Datalog as its policy language. Its certificate retrieval system provides the possibility of querying remote systems given their IP addresses. In addition, it creates a proof that validates that the answer is correct. The RT framework is a set of languages for representing policies and credentials where attributes are represented as roles. Starting with the most basic language, called RT_0, each new one provides extra features. Cassandra is also a role-based trust management system. It uses a policy language based on Datalog with constraints, and its expressiveness can be adjusted by changing the constraint domain. It permits the specification of permissions, delegation of authority, revocation, and access control. Bonatti and Samarati describe an approach for specifying information disclosure constraints and the inference process necessary to reason over them and filter relevant policies given a request. PeerTrust provides most of the features that the previous systems include, such as query facilities, a Datalog-based policy language, trust negotiation capabilities, and delegation of authority. Additionally, the PeerTrust language adds private and signed rules and references to issuers and requesters, allowing more expressive policies. PeerTrust also adds the possibility of nesting remote credential requests and disclosures and, like the SD3 system, generates a proof with each answer that validates it.

For further reading, we encourage you take a look at the Working Group I2 (on policy language, enforcement, and composition) that the EU/IST FP6 Network of Excellence REWERSE has established (http://rewerse.net) and the TrustBuilder homepage (http://dais.cs.uiuc.edu/trustbuilder/index.html). You can also participate in the forthcoming Workshop on Trust, Security, and Reputation on the Semantic Web at the 2004 International Semantic Web Conference (http://trust.mindswap.org/trustWorkshop).

Acknowledgments

Our research was partially supported by the projects ELENA (www.elena-project.org, IST-2001-37264) and REWERSE (http://rewerse.net, IST-506779).

References

1. W. Nejdl et al., "EDUTELLA: A P2P Networking Infrastructure Based on RDF," *Proc. 11th Int'l World Wide Web Conf.* (WWW 2002), ACM Press, 2002, pp. 604–615.

2. A. Halevy et al., "Piazza: Data Management Infrastructure for Semantic Web Applications," *Proc. 12th Int'l World Wide Web Conf.* (WWW 2003), ACM Press, 2003, pp. 556–567.

3. T. Yu, M. Winslett, and K. Seamons, "Supporting Structured Credentials and Sensitive Policies through Interoperable Strategies in Automated Trust Negotiation," *ACM Trans. Information and System Security*, vol. 6, no. 1, 2003, pp. 1–42.

4. R. Gavriloaie et al., "No Registration Needed: How to Use Declarative Policies and Negotiation to Access Sensitive Resources on the Semantic Web," *Proc. 1st European Semantic Web Symp.*, Springer-Verlag, 2004, pp. 342–356.

5. J. Basney et al., "Negotiating Trust on the Grid," *Proc. 2nd Workshop Semantics in Peer-to-Peer and Grid Computing*, 2004, www.isi.

edu/~hongsuda/SemPGRID04/proceedings/proceedings.pdf.

6. J. Trevor, "SD3: A Trust Management System with Certified Evaluation," *Proc. IEEE Symp. Security and Privacy*, IEEE CS Press, 2001, p. 106.

7. N. Li and J.C. Mitchell, "RT: A Role-Based Trust-Management Framework," *Proc. DARPA Information Survivability Conf. and Exposition* (DISCEX 03), IEEE CS Press, 2003.

8. M.Y. Becker and P. Sewell, "Cassandra: Distributed Access Control Policies with Tunable Expressiveness," *Proc. IEEE Workshop Policies in Distributed Systems and Networks* (POLICY 2004), IEEE CS Press, 2004, pp. 159–168.

9. P.A. Bonatti and P. Samarati, "A Uniform Framework for Regulating Service Access and Information Release on the Web," *J. Computer Security*, vol. 10, no. 3, 2002, pp. 241–272.

Trust, but Verify: Emergence, Trust, and Quality in Intelligent Systems

Vipul Kashyap, *Clinical Informatics R&D, Partners Healthcare System*

Trust, but verify—the late Ronald Reagan (US president, 1980–88) enunciated this well-known political dictum in his attempts to break down barriers and integrate communist systems (political, economic, and nuclear) into a capitalist world. This dictum is equally appropriate in the context of information integration and service composition for creating and deploying intelligent systems. Biomedical information and research is a representative domain that presents complex and interesting challenges. It's also a prime candidate for applying semantics-based approaches.

Biomedical data integration challenges

Biomedical and biological research has grown from a cottage industry marked by scarce, expensive, manually generated data to a large-scale, data-rich industry marked by factory-scale sequencing. Biomedicine is fast becoming an information-based science, with data and information playing a big role across the research flow (for example, genomics → transcriptomics → proteomics → metabolomics → final products and results). Scientific-data integration is one of the most daunting challenges at the interface between computer science and biology[1] and has been seen as restraining rapid progress in biomedical research (see http://nihroadmap.nih.gov/bioinformatics/index.asp. Biomedical data integration poses a unique set of challenges:[1–3]

- *Diversity of information objects, including data types and queries*: sequences, complex phenotypic and disease-relevant data, graphs, 3D structures, and images; and similarity-based queries (for example, sequence similarity), classification-based queries (for example, papers about gene X), and what-if hypotheses-generating queries (what if gene X was suppressed—will protein Z exist?)
- *Diversity of information-based computations and operations*: experimental plans and protocols using complex repetitive computations involving data retrieval, fusion, and analytics; data curation and annotation tasks; and hypothesis validation across multiple requiring scenarios and federated search-query processing
- *Semantic heterogeneity*: multiple controlled vocabularies and ontologies requiring integration, interoperation, and composition; and semiautomatic creation and verification of mappings, including complex mappings across vocabularies and ontologies; and mappings and annotations of data objects to ontological concepts
- *Biomedical research's dynamic and evolving nature*: evolution of schemas, ontologies, and vocabularies and their impact on underlying mappings or annotations; evolution of data objects and their impact on associated mapping and annotations; data uncertainty and inconsistency; and support for proactive data mining and hypothesis generation

Biomedical information and research is a representative domain that presents complex and interesting challenges. It's also a prime candidate for applying semantics-based approaches.

The Semantic Web effort[4] attempts to address some of these problems by explicitly capturing the semantics of information and services in a machine-readable format (see www.w3.org/RDF and www.w3.org/TR/owl-features). Meanwhile, over and above the scale of the data and information explosion, semantics itself is a moving target that keeps evolving with time and use. This calls into question the formal top-down and classical logic-based approaches various Semantic Web researchers are proposing. An alternative and potentially better-suited approach that's bottom-up and statistical in nature investigates the phenomena of *emergence* in the context of information and service semantics.[3,5–8] (This proposition itself is worthy of debate but isn't the focus of this essay.)

The critical question that arises is, how do we guide this process? That is, how do we ensure that the emergent processes don't generate spurious and flawed information? As the emergent semantics process flow (which I discuss later) goes through multiple iterations, it generates various artifacts (taxonomies, ontologies, mappings, and so on). The quality of the data and information these artifacts represent can help guide these emergent processes. However, early on the process will probably be in a bootstrapping mode, so you need an a priori measure of information quality. I'd argue here that an *a priori trust* in the artifact component or source is the only way this is possible. However, as the system creates and evolves more semantics, we must modulate this a priori measure with something that measures the generated artifacts' internal consistency and validity. That is, we must verify the a priori measures—hence the dictum: trust, but verify.

An emergent semantics framework for biomedical informatics

An emergent semantics-based information infrastructure,[3] which Figure 4 shows, would be a proactive platform where people and applications could collaborate to create dynamic semantics reflecting the current state of knowledge in biomedical research. This infrastructure would have some interesting properties:

- *Self-description.* The infrastructure would enable self-description of biological data and content. This is the focus of XML-

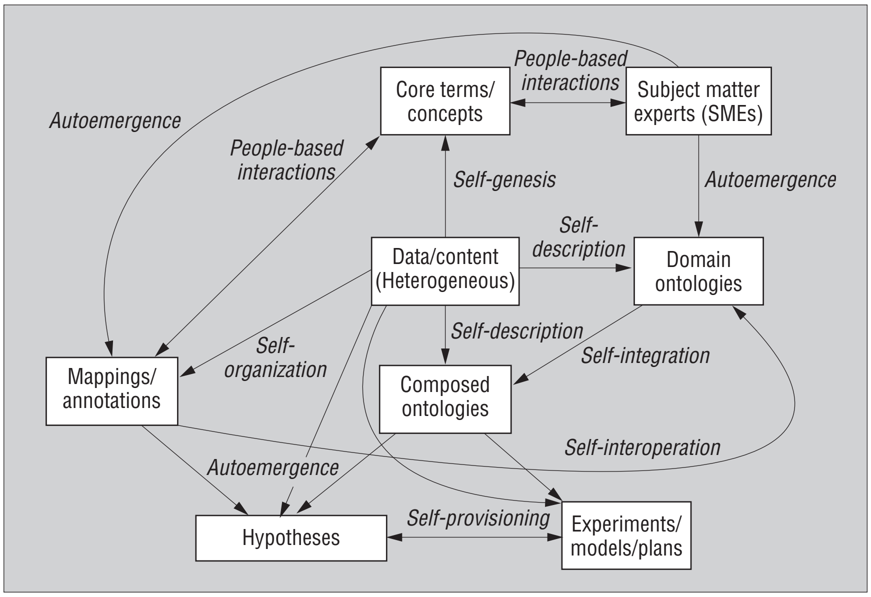


Figure 4. A strawman emergent semantics infrastructure and process.

based markup languages—for example, BioPAX (www.biopax.org) and OWL—and ontologies or vocabularies—for example, GeneOntology (www.geneontology.org) and the Unified Medical Language System Semantic Network.[9]

- *Self-genesis*. The infrastructure would proactively analyze data and content flowing through it to create models, ontologies, and concepts to capture semantics, enabling bootstrapping of prevalent meanings.
- *Autoemergence*. The infrastructure would monitor interactions between people and applications to capture and describe new meanings and knowledge that emerge, or existing meanings and knowledge that evolve from these interactions. This property might enable proactive generation of new biomedical hypotheses on the basis of new emerging knowledge.
- *Self-organization*. The infrastructure would (re)organize itself to perform new tasks in response to new requests for information and services or due to new meanings emerging. Two types of self-organization are self-interoperation or integration, and self-provisioning. With the former, the infrastructure would integrate or interoperate between existing meanings in response to new requests for information and services. This might lead to creating data annotations or mappings between concepts in ontologies. With the latter, the infrastructure would monitor data retrieval and computation operations invoked and proactively create experimental plans, workflows, protocols, and models.

In the initial bootstrapping stages, where little information is available about the various components, we must depend on a priori quality measures.

Emergence, information quality, and trust

We can classify the collection of artifacts or components that the emergent semantics process generates (see Figure 4) under two broad categories: *source components* present at the beginning of the process and *derived artifacts* generated during the process. Source components include data and content sources and subject matter experts (SMEs). Derived artifacts include domain ontologies, both single and composed; annotations and mappings; hypotheses; and experiments, models, and plans.

The generated artifacts must be of good quality. "Quality" might refer to both the information and data quality of domain ontologies, annotations, mappings, and data sources and the quality of service related to software components that you might develop to realize various plans, experiments, and so on. Quality metrics for the artifacts I just discussed fall into two broad categories:

- Intracomponent or artifact quality measures, including those that evaluate quality on the basis of some internal consistency criteria and those assigned a priori to an artifact
- Intercomponent quality measures, including those that come from quality measures assigned to other related components

In the initial bootstrapping stages, where little information is available about the various components, we must depend on a priori quality measures. Furthermore, trust metrics are highly correlated with these quality measures, at least initially. Finally, over time, we should adjust quality metrics to consistently reflect external components' quality measures. Trust, but verify.

Colleagues and I at the Dagstuhl Workshop on Data Quality discussed one possible approach for generating a joint measure or metric to reflect this interrelationship.[10] We propose pragmatic and workable definitions (and not philosophical ones) for the notion of trust and other concepts it depends on. Figure 5 shows a proposed taxonomy and definition flow chart. Verifying the data and information captured in an artifact or consumed or produced by the service a component provides forms the organizing principle of the taxonomy in Figure 5. We can organize various notions of trust around this notion of verifiability.

Fact

If we can verify data or information by some means, it's a fact. Verifiability might be based on either established theories (3 + 2 = 5) or on trust in an authority (the US Government Web site currently states that George W. Bush is the US president). If we can't verify the information stored in the data by some means, we must have belief in that piece of information.

Belief

When we assume that some piece of data or information or a service's quality is valid without objective verification or recourse to a trusted authority, it's a belief. According to the *Merriam-Webster Online Dictionary*, belief is "a state or habit of mind in which trust or confidence is placed in some person or thing," a notion intimately connected to trust.

A priori belief

When you assume the validity of some data or information in an a priori manner, without an underlying rationale (bordering on faith), it's an a priori belief. This notion appears biblical, but if we analyze how intelligent systems are used or deployed we might find that it's used quite often in this field.

Rational belief

When we assume some data or information's validity on the basis of some rationale or justification, it's a rational belief. An example is an intelligent system's reputation, which is based on past experiences with the system. (In the emergent semantics process flow in Figure 4, reputation-based belief is relevant in the context of SMEs, a component in the process flow.) We might measure these past experiences objectively and empirically. For instance, we might use measures such as consistency, completeness, or quality of service typically used in the data quality literature to characterize past results received from an information system.

Trust

The notion of trust is based on the continuum of belief notions I just defined. So, we might have a priori trust in some information, data, or service. This a priori trust could be useful in the initial bootstrapping stages of the emergent semantic process flow and could be modulated with more objective measures such as consistency, completeness, and quality of service. Finally, trust could either be direct (I trust this information) or indirect or transitive (I trust this information because my friends trust it).[11] In the latter case, we can assume that trust decays in proportion to the length of the "trust chain."

The trust-quality metric

We can view the trust-quality metric as a multidimensional measure. One set of dimensions could describe objective (or verifiable) measures, and another could comprise subjective (or a priori) measures. We could combine dimensions in two ways.

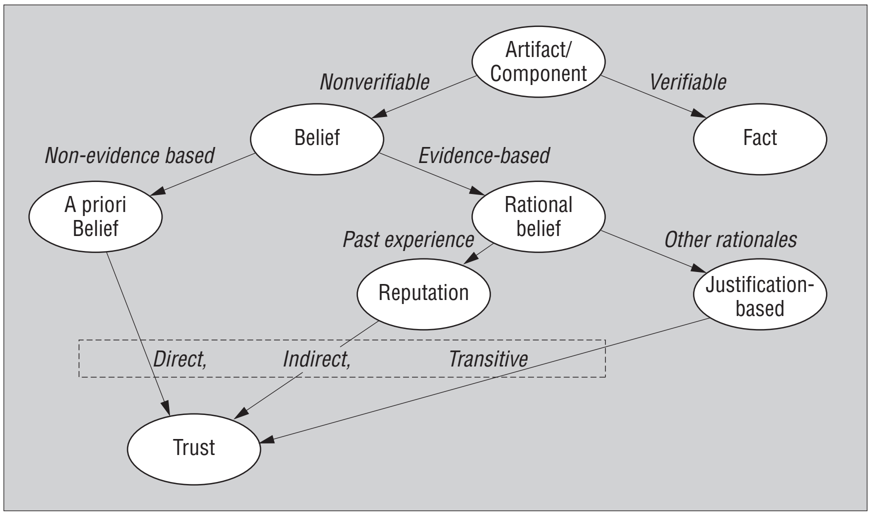


Figure 5. A taxonomy of fact, belief, and trust.

Multidimensional comparison

We might prioritize the dimensions, according a higher priority to the objective ones. So, a comparison function could first choose to compare the objective dimensions. In a case where the comparison function based on the objective dimensions returns values that are equal or within a statistical error range, we could then compare the subjective dimensions.

Composition into a single measure

The other approach is to combine the dimensions' values into a single common measure by assigning weights to the various dimensions. For instance, we could combine a subjective measure such as a priori belief and an objective measure such as reputation as

$$\text{Quality} = \alpha \times \text{AprioriBelief} + \beta \times \text{Reputation}$$

The weights might reflect application requirements. For instance, in the absence of any other information, we might want to set $\alpha = 1$, $\beta = 0$. Furthermore, a priori belief itself might have varying levels of subjectivity, and reputation might be characterizable in terms of information or data quality factors, such as consistency and completeness, or quality-of-service factors:

$$\text{Quality} = \alpha_1 \times \text{DirectTrust} + \alpha_2 \times \text{IndirectTrust} + \beta_1 \times \text{Consistency} + \beta_2 \times \text{Completeness} + \beta_3 \times \text{QualityOfService}$$

Of course, we must do more experiments to refine these empirical equations and understand the true nature of the interrelationship between trust and quality. Till then, we'll have to trust, but verify! ■

References

1. "Digital Biology: The Emerging Paradigm," *Biomedical Information Science and Technology Symp.*, 2003.
2. "Workshop on Data Management for Molecular and Cell Biology," 9 Dec. 2003, http://pueblo.lbl.gov/~olken/wdmbio.
3. V. Kashyap, "Emergent Semantics: An Organizing Principle for Biomedical Informatics and Knowledge Management," *Semantic Web Challenges for Knowledge Management*, *AIS SIGSEMIS Bulletin*, vol. 1, no. 2, 2004.
4. T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web: A New Form of Web Content That Is Meaningful to Computers Will Unleash a Revolution of New Possibilities," *Scientific American*, May 2001.
5. C. Behrens and V. Kashyap, "The 'Emergent' Semantic Web: A Consensus Approach for Deriving Semantic Knowledge on the Web," *Real World Semantic Web Applications, Frontiers in Artificial Intelligence and Applications*, vol. 92, IOS Press, 2002.
6. A. Mädche et al., "Emergent Semantics," *IEEE Intelligent Systems*, vol. 17, no. 1, 2002, pp. 78–86.
7. K. Aberer et al., "Emergent Semantics Systems," *Proc. Int'l Conf. Semantics of a Networked World: Semantics for Grid Databases*, 2004.

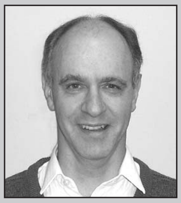
The Authors

Bharat Bhargava is a professor with the Department of Computer Sciences and the Center for Education and Research in Information Assurance and Security at Purdue University. Contact him at bb@purdue.edu.

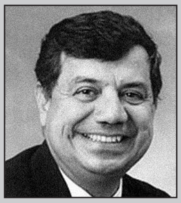
Leszek Lilien is a research associate with the Department of Computer Sciences and the Center for Education and Research in Information Assurance and Security at Purdue University. Contact him at llilien@purdue.edu.

Arnon Rosenthal is a principal scientist at the MITRE Corp. Contact him at arnie@mitre.org.

Marianne Winslett is a research professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign. Contact her at winslett@uiuc.edu.

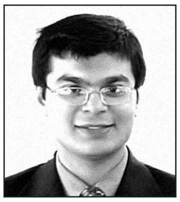


Morris Sloman leads the Distributed Software Engineering Group in the Department of Computing, Imperial College London. Contact him at the Dept. of Computing, Imperial College London, 180 Queen's Gate, London SW7 2AZ, UK; m.sloman@imperial.ac.uk; www.doc.ic.ac.uk/~mss.



Tharam S. Dillon is the Dean of Faculty of Information Technology at the University of Technology, Sydney. He is also the foundation professor of computer science, computer engineering, and software engineering and the director of the eXel research group. Contact him at tharam@it.uts.edu.au.

Elizabeth Chang is a professor in IT and software engineering at the Curtin University of Technology and the director of the Centre of Excellence for Extended Enterprises and Business Intelligence research group. Contact her at change@cbs.curtin.edu.au.



Farookh Khadeer Hussain is a PhD student at the Curtin University of Technology's School of Information Systems. Contact him at hussainf@cbs.curtin.edu.au.

Wolfgang Nejdl is a professor at the University of Hanover's Information Systems Institute and Computer Science Department, and the director of the L3S Research Center. Contact him at nejdl@l3s.de.

Daniel Olmedilla is a PhD student at the University of Hanover's Computer Science Department and a research scientist at the L3S Research Center. Contact him at olmedilla@l3s.de.

Vipul Kashyap is a senior medical informatician in the Clinical Informatics Research & Development group at Partners HealthCare System. Contact him at Partners HealthCare System, Clinical Informatics R&D, 93 Worcester St., PO Box 81905, Wellesley, MA 02481; vkashyap1@partners.org.

8. K. Aberer, P. Cudré-Mauroux, and M. Hauswirth, "The Chatty Web: Emergent Semantics through Gossiping," *Proc. 12th Int'l World Wide Web Conf.* (WWW 2003), ACM Press, 2003.

9. V. Kashyap and A. Borgida, "Representing the UMLS Semantic Network Using OWL (or What's in a Semantic Web Link?)," *Proc. 2nd Int'l Semantic Web Conf.* (ISWC 03), LNCS 2870, Springer-Verlag, 2003, pp. 1–16.

10. S. Boll et al., "Do You Trust in Data Quality?" *Working Group on Trust and Data Quality, Dagstuhl Workshop on Data Quality*, Nov. 2003; http://sirius.cs.ucdavis.edu/Dagstuhl03/wgs/03362.KashyapVipul.Other.ppt.

11. M. Richardson, R. Agrawal, and P. Domingos, "Trust Management for the Semantic Web," *Proc. 2nd Int'l Semantic Web Conf.* (ISWC 2003), LNCS 2870, Springer-Verlag, 2003, pp. 351–368.