Using Trust for Secure Collaboration in Uncertain Environments

The SECURE project investigates the design of security mechanisms for pervasive computing based on trust. It addresses how entities in unfamiliar pervasive computing environments can overcome initial suspicion to provide secure collaboration.

Vinny Cahill, Elizabeth Gray, Jean-Marc Seigneur, Christian D. Jensen, and Yong Chen
Trinity College Dublin

Brian Shand, Nathan Dimmock, Andy Twigg, and Jean Bacon
University of Cambridge

Colin English, Waleed Wagealla, Sotirios Terzis, and Paddy Nixon
University of Strathclyde

Giovanna di Marzo Serugendo and Ciarán Bryce
University of Geneva

Marco Carbone, Karl Krukow, and Mogens Nielsen
University of Aarhus

Registered parties behind firewalls in strictly controlled environments carry out most substantial, accountable computation. However, pervasive computing foresees a massively networked infrastructure supporting a large population of diverse but cooperating entities. Entities will be both autonomous and mobile and will have to handle unforeseen circumstances, ranging from unexpected interactions with other entities to disconnected operation.

This infrastructure introduces new security challenges that existing security models and mechanisms don't adequately address. Because of the infrastructure's scale, the security policy must encompass billions of potential collaborators. Mobile entities will often become disconnected from their home networks and must be able to make fully autonomous security decisions; they can't rely on specific security infrastructures such as certificate authorities and authorization servers. Although certificate authorities might help establish other collaborators' identities, in the environment envisaged, identity conveys no a priori information about a principal's likely behavior. Because of the infrastructure's dynamism, entities that offer services will be confronted with requests from unknown entities, and mobile entities will need to obtain services in unfamiliar, possibly hostile environments. A party facing such a complex world stands to benefit from interaction, but only if it can respond to new entities and assign meaningful privileges to them.

The Secure Environments for Collaboration among Ubiquitous Roaming Entities (SECURE) project is designing a novel security approach that addresses these challenges. If successful, this approach will significantly benefit not only future systems but also various emerging mobile computing applications. It could also benefit collaborations over the Internet where correspondents' identities and intentions are difficult to establish with certainty.

Our approach applies the human notion of trust. This naturally leads to a decentralized security management approach that can tolerate partial information, albeit one that has inherent risks for the trusting entity. Fundamentally, the ability to reason about trust and risk is what lets entities accept risk when interacting with other entities. The SECURE project seeks a formal basis for reasoning about trust and risk and for deploying verifiable security policies, embodied in a computational framework that is adaptable to various application scenarios.

For example, consider the problem of routing messages in an ad hoc wireless network. An entity,

Published by the IEEE CS and IEEE ComSoc ■ 1536-1268/03/$17.00

or mobile node, with a message to send must rely on other nodes on the path to the intended destination to forward its message. Generally, the intermediate nodes might have no a priori relationship or agreement with the sender, which they might never have encountered before. Also, forwarding messages costs battery and processing power. Why should a sender rely on such nodes to help it? If multiple paths exist, which path should the sender put the most confidence in? These *trusting decisions* are informed by the degree to which the sender trusts the intermediate nodes to do the right thing based on observations of and experience with these nodes, their reputations, and possibly recommendations from third parties. These decisions are also mediated by the risk the sender takes. The sender probably requires less trust to send a low-importance message and more trust for a very important message that really needs to get through.

Understanding trust

Humans use trust daily to promote interaction and accept risk in situations where they have only partial information. Trust lets one person assume that another will behave as expected. Despite the extensive study of trust in sociology, psychology, and philosophy, it remains an elusive concept that defies stringent definition. This is partly because trust is largely invisible and implicit in society. Various definitions of trust have been offered,[1] many of which depend on the author's viewpoint or the context in which he or she examines trust. Because of trust's multifaceted nature, it's difficult to form a unified definition.

It's useful to examine dictionary definitions of trust to determine which are widely accepted. Common to these definitions are the notions of confidence, belief, faith, hope, expectation, dependence, and reliance on the integrity, ability, or character of a person or thing. The variety of common terms shows that there is no precise definition and hints at the range of views of trust. Sociologist Diego Gambetta[2] introduces trust as "a particular level of the subjective probability with which an agent assesses that another agent or a group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action." Social psychologist Morton Deutsch's work[3] considers trust when faced with an ambiguous path with beneficial or harmful results depending on another person. He identifies various types of trust, from trust as the fallback when no other option is available to trust as confidence that the desired outcome will be reached. Deutsch suggests that people take trusting actions when possible benefits outweigh the likelihood of being let down. This implies that risk analysis forms an important part of the trust decision. Due to these and other views, Stephen Marsh[4] reasons that it might prove more suitable to model trust's behavior rather than trust itself, removing the need to adhere to specific definitions.

An important observation from all these sources is that trust—one individual's opinion of another—is a subjective notion, and every individual decides whether to trust based on the evidence available for personal evaluation (although you might delegate this decision to a more authoritative source in certain circumstances). Also, trust isn't symmetric—two individuals don't need to have similar trust in each other. Even if two entities get the same evidence, they might not necessarily interpret this information in the same way.

Trust is also situation-specific; trust in one environment doesn't directly transfer to another environment. So a notion of context is necessary. Despite this situational nature, there's some agreement on a dispositional aspect of trust as a measure of your propensity to believe in others' trustworthiness.

Social scientists also highlight trust's dynamic properties: It is self-preserving and self-amplifying, it increases through periodic successful interactions, and it degrades through disuse or misuse.

Trust is situation-specific; trust in one environment doesn't directly transfer to another environment. So a notion of context is necessary.

Trust is inherently linked to risk; there's no reason to trust if there's no risk involved. This relationship implies that cooperation is less likely with higher risk unless the benefits from cooperating are worth the risk. So reasoning about trust lets entities accept risk when interacting with others.

The SECURE project's approach is based on the premise that trust and risk are inexorably linked and must both be considered when making a decision about an ambiguous path whose outcome depends on another entity's actions.

Handling trusted interactions

The trust a principal needs for an interaction depends on the risk involved. This allows for appropriate security in pervasive environments without requiring excessive trust in straightforward cases.

When a system grants privileges to a principal, it expects the principal to use

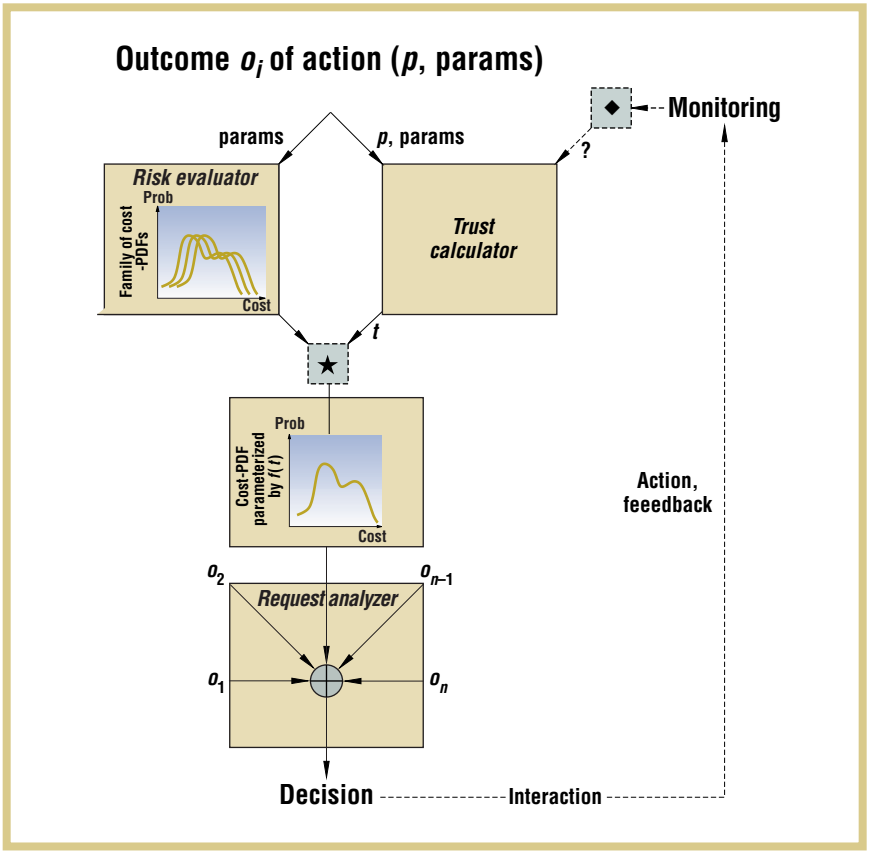

Figure 1. Decision making using trust and risk.

them in a particular way—for example, to update old address book entries with accurate information. However, the principal could deviate from this expected behavior, and the combined likelihood and severity of this is the risk of granting them a privilege.

Risk analysis

In SECURE, the risks of a trust-mediated action are decomposed by possible outcomes. Each outcome's risk depends on the other principal's trustworthiness (the likelihood) and the outcome's intrinsic cost. For example, an address update might itself be out-of-date or maliciously misleading. These two outcomes' costs would reflect the user's wasted time, and the likelihoods would depend on trust in the other party.

An outcome's costs could span a range of values. For example, a user might have received a correct phone book entry. This third outcome's cost could show a net benefit to the user, as the user might successfully use it later. However, if the number became out-of-date by the time it was used, that would be a net loss. To reflect this uncertainty, you might represent the distribution of costs as a cost-PDF (probability density function).

Figure 1 illustrates a user contemplating a parameterized interaction with principal p. For each possible outcome, the user has a parameterized cost-PDF (a family of cost-PDFs) that represents the range of possible costs and benefits the user might incur should each outcome occur.

While the *risk evaluator* assesses the possible cost-PDFs, the *trust calculator* provides information t that determines the risk's likelihood based on the principal's identity p and other parameters of the action. The risk evaluator then uses this trust information to select the appropriate cost-PDF.

Finally, the *request analyzer* combines all the outcomes' cost-PDFs to decide if the action should be taken or to arrange further interaction. Because any uncertainty is preserved right up to the decision point, this allows more complex decision making than simple thresholding, allowing responses such as "not sure" if there isn't enough information.

In our continuing example, if Liz's PDA received a phone number from Vinny's PDA, she might not think it was maliciously misleading based on her trust in Vinny's honesty. She might think it could be out-of-date, however, if Vinny had given her stale information before, attributing a higher risk to this outcome. Finally, she'd consider the potential benefit of having a correct number, again moderated by Vinny's trustworthiness. Liz's PDA would do all these calculations on her behalf using its model of her trust beliefs, as Figure 2 illustrates. If the benefits outweighed the other outcomes' costs, the PDA would then accept the information.

On the other hand, if John—a colleague from a competing research group—sent Liz an address book entry, her PDA might reject it after the same analysis because she didn't know him. At this point, the request analyzer might seek out more information, maybe by discovering that John works with Jean, who is trusted by Liz, or by interrupting Liz for confirmation.

Interconnected address book categories can give structure to information.[5] By assigning each category a risk value and explicitly costing the user's time, sensitive entries (such as bank phone numbers) can naturally be protected with little user effort.

So this explicit risk analysis, which differentiates the SECURE approach from other trust-based approaches,[6,7] balances the evidence that a principal is

trustworthy against the risks if it isn't. This allows sensible behavior in the face of uncertainty, but prevents abuse by incrementally updating trust assessments as more evidence becomes available.

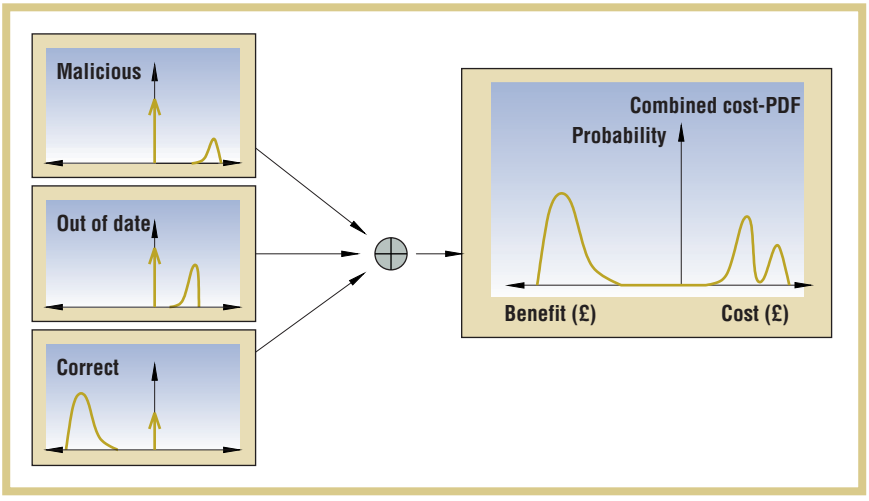

Figure 2. Illustration of risk analysis.

Building trust

Recent work on trust management systems[8–11] attempts to manage security in large-scale distributed networks by using credentials that delegate permissions. However, these systems focus on trust management's static element and neglect the dynamic component of trust formation. What does trust really consist of?

Fundamentally, we base trusting decisions on trust information, encompassing evidence from personal observations of previous interactions and recommendations from partly trusted third parties. These two main sources of stored trust information let us dynamically form an opinion about another entity.

Personal observations of the entity's behavior, through recording outcomes of interactions, are essential in subjectively evaluating trustworthiness. These observations are evaluated against the principal's expected behavior to produce *experiences*.[12] The range of experience values reflects the effect of the observed outcome relative to the expected outcome, usually in terms of gain or loss. These values are ordered and classified into two sets, a trust-positive set and a trust-negative set. This evidence is aggregated with the evidence from previous interactions to give a comprehensive summary of the principal's interaction history.

Recommendations from trustworthy third parties can propagate trust in unknown entities by providing supporting evidence for decisions. The recommendation process becomes more important when the trust evaluation based on observations isn't precise enough. In such cases, an entity might need more information. Also, it could discard imprecise recommendations that provide little additional information. The decision, however, is left to the individual entity.

Upon receiving a collaboration request, we can dynamically filter the available trust evidence to keep only what's relevant to the requested action. If no evidence from experience or recommendation is available for an entity, we must establish an initial trust value to encourage low-risk collaborations. We can determine this using many strategies.[13] This collaboration will provide further evidence on which we can base future trust formation. If enough evidence exists to reason about the entity's trustworthiness, then we will evaluate observations and recommendations to yield trust information. This information might be multidimensional—separate trust intervals might be formed for different aspects of trust in the interaction. We treat recommendation evidence separately from personal experience evidence, which has more influence on trust.

The trust model (see the related sidebar) operates using local trust policies. These local policies let the system use collected evidence and dictate the conditions in which the trust opinion, formed from the evidence, should be used. The policies also let us conditionally delegate trust evaluation to an outside entity—an important feature of the trust model. The difference between delegation and recommendation is that we delegate to entities similar to ourselves, which we might consider experts for the decision; in recommendation, however, we gather trust information from any principal in the environment and can seek more than one recommendation. We can also weigh recommendations according to our trust in the source as a recommender.

Our framework goes a step further than trust-based frameworks such as Confidant[14] and similar approaches.[15] Confidant, for instance, is designed for ad hoc network routing. No centralized and trusted network manager exists in such a network; each node must trust others to transmit its messages. Nodes can exchange reputation information to detect malicious nodes. A node might send *reputation records* that describe its first-hand experience with another node or *trust records* that encapsulate reputation information received from other nodes. Nodes depend on these records to make routing decisions. In our approach, a node's trust decision need not rely on exchanged reputation information but can also be delegated to another node. This leads to a more flexible range of trust policies and is more consistent with the human approach to trust formation.

The Trust Model

The trust model[1] aims to provide formal techniques for studying the properties of trust-based systems. Our formal model focuses on the set $\mathcal{T}$, the set of trust values, whose elements represent degrees of trust. The set $\mathcal{T}$ has two orderings $\preceq$ and $\sqsubseteq$ such that $(\mathcal{T}, \preceq)$ is a complete lattice and $(\mathcal{T}, \sqsubseteq)$ is a complete partial order with a bottom element. The ordering $\preceq$ reflects the notion of "more trust" saying that a particular trust value might represent a higher level of trust than another, whereas the ordering $\sqsubseteq$ reflects information saying that a particular trust value might contain more information than another. In a setting with numerous interacting principals, we can't assume that every principal has precise information, or even any information at all, about every other principal. Principals must often act on requests from unknown principals. The element $\bot_{\sqsubseteq}$ represents the value `unknown`. In this setting, it's important to distinguish this element from $\bot_{\preceq}$, which represents no trust; the former is interpreted as having no evidence for trust or distrust, whereas the latter implies an explicit reason for distrusting the particular principal.

We have a technique[1] for building the triple $(\mathcal{T}, \preceq, \sqsubseteq)$ starting with a complete lattice, $(D, \leq)$, and considering the set of intervals of type $[d_0, d_1]$ over D. The ordering $\preceq$ will be a natural lifting of $\leq$ to these intervals. The $\sqsubseteq$ order considers the intervals' "width," which can be thought of as representing the amount of uncertainty. So the real trust value is in that interval, but we aren't sure exactly where.

A simple example could be starting with the complete lattice of reals ($[0, 1], \leq$). This lattice's intervals are the subintervals of [0,1] (where [0,1] denotes complete uncertainty). The interval [.3, .6] could represent the trust principal a has in principal b with uncertainty $0.6 - 0.3 = 0.3$. a might eventually receive more information about b, enabling it to give a more precise judgment—narrowing the value to [.33, .6], for example.

Given a set of principals $\mathcal{P}$ and the set $\mathcal{T}$, we can see trust information as a function:

$$m : \mathcal{P} \to \mathcal{P} \to \mathcal{T}.$$

The function m applied to a applied to b is the trust value $m(a)(b) \in \mathcal{T}$ expressing a's trust in b.

Furthermore, every principal has a local policy that is its contribution to the global trust information. It expresses how the principal plans to compute trust information. The model can handle delegation, which means that a principal can refer to another principal's trust information. So every principal can express its trust in another principal in terms of not only its own beliefs but also other principals' beliefs. Given $a \in \mathcal{P}$, the policy π_a can be seen as a function of the type:

$$\pi_a : (\mathcal{P} \to \mathcal{P} \to \mathcal{T}) \to (\mathcal{P} \to \mathcal{T}).$$

This function takes the current trust information about all principals (function m) and returns a function which expresses a's trust in a given principal.

The collection of all local policies induces a global policy function:

$$\Pi : (\mathcal{P} \to \mathcal{P} \to \mathcal{T}) \to (\mathcal{P} \to \mathcal{P} \to \mathcal{T})$$

(The induced function is $\Pi' : \mathcal{P} \to (\mathcal{P} \to \mathcal{P} \to \mathcal{T}) \to (\mathcal{P} \to \mathcal{T})$, which is equivalent to the given one.)

As an important general assumption, we require that Π is a $\sqsubseteq$-continuous function. Intuitively, this is a reasonable assumption: All policies should satisfy the property that the more information other principals provide, the more information the policies provide. From this assumption, we safely define the global trust information as the least fixed point of Π.

REFERENCE

1. M. Carbone, M. Nielsen, and V. Sassone, *A Formal Model for Trust in Dynamic Networks*, BRICS tech. report RS-03-4, Univ. Aarhus, 2003.

Software framework

Even if we understand how to reason about trust formation and evolution and how to exploit trust in making access control decisions, we also need to ensure that we can feasibly implement the necessary algorithms for these processes in heterogeneous systems. So, we are developing a policy-neutral software architecture framework encompassing algorithms for trust management that we can use in various applications. Figure 3 illustrates the current version of our framework design.

When a principal p makes a request for interaction, the request passes through the application programming interface into the request analyzer. The request analyzer requests information about p from three sources: the *entity recognition* component, the trust calculator, and the risk evaluator.

The entity recognition component, which is responsible for recognizing new or previously encountered entities, requests verification that p is recognized (see the "Entity Recognition" sidebar). Any other component can consult the entity recognition component to obtain recognition capabilities as necessary.

Additionally, the request analyzer requests a trust calculation from the trust calculator. The trust calculator computes the least fixed point, as we discuss in "The Trust Model" sidebar, using information gathered from the *trust lifecycle*

management component and its local trust policy. As we mentioned earlier, the system can delegate the trust level calculation for *p* to another entity, thereby initiating synchronous communication with a remote entity.

The trust calculator's local policies update on the basis of information fed from the trust lifecycle management component. This component handles trust formation, evolution, and exploitation on the basis of data drawn from the *evidence store*. The trust lifecycle management policy allows trust information to be weighted according to context-specific criteria.

The evidence store holds all trust- and risk-related data. It is updated with data from evidence gathering, such as recommendations and security updates collected in an asynchronous process, and from the *monitoring* component. The evidence store also responds to requests for recommendations from other entities.

The monitoring component observes actual interaction with *p* and conveys the results of the interaction to the evidence store.

The request analyzer also requests a risk assessment from the risk evaluator, which calculates the request's potential risk based on the local information stored in the *risk configuration* component, which is updated with information from the evidence store.

The system assesses and aggregates all of the information it obtained about *p*.

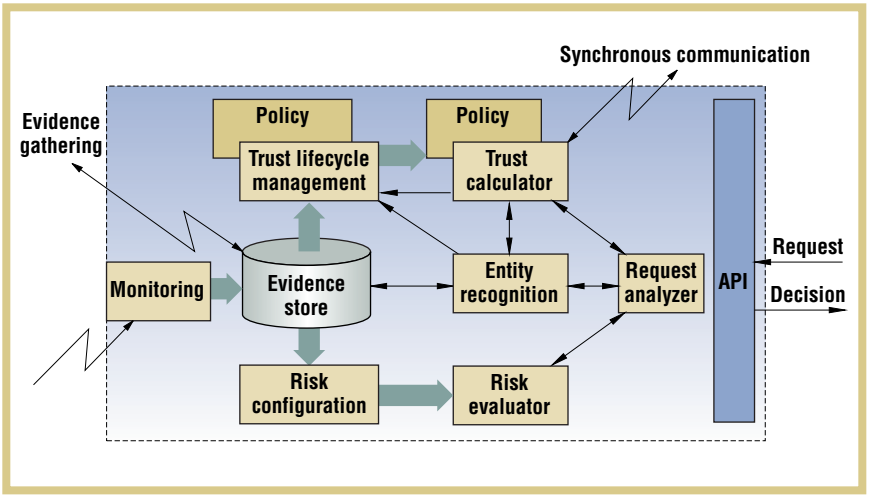

Figure 3. The framework for one party to an interaction. Large arrows signify data flows and thin arrows signify control flows.

Entity Recognition

Authentication in pervasive computing systems isn't necessarily enough to ensure security because identity conveys no a priori information about the other entity's likely behavior.[1,2] Entity recognition,[2] which doesn't bind an externally visible identity to the recognized entity like authentication does, has been proposed as a more general replacement for authentication. We suppose that the ability to reliably recognize another entity is sufficient to establish trust in that entity based on past experiences, and entity recognition provides a local reference for trust, which is in turn maintained by other components in the SECURE (Secure Environments for Collaboration among Ubiquitous Roaming Entities) framework.

The SECURE framework includes an entity recognition component based on Pluggable Recognition Modules, which allows the integration of more or less secure recognition schemes (such as traditional authentication modules developed for PAM [Unified Login with Pluggable Authentication Modules][3]) or pure recognition-based schemes (such as APER [A Peer Entity Recognition],[2] which uses signed claims broadcast periodically on a network to recognize entities). A particular recognition scheme's accuracy must be assessed and a level of confidence associated with the outcome of the recognition process. For example, the average attack space[4] of recognition schemes could give an upper bound for the confidence in a particular recognition scheme. APER provides three levels of confidence depending on how much verification is applied to the claims: signature validation, claim freshness, and challenge-response.

REFERENCES

1. S. Creese et al., "Authentication for Pervasive Computing," *Proc. 1st Int'l Conf. Security in Pervasive Computing*, IEEE CS Press, 2003.
2. J.M. Seigneur et al., *End-to-end Trust Starts with Recognition*, tech. report TCD-CS-2003-05, Computer Science Dept., Trinity College Dublin, 2003.
3. V. Samar and R. Schemers, "United Login with PAM," Open Software Foundation, 1995, www.opengroup.org/tech/rfc/rfc86.0.html.
4. R.E. Smith, *Authentication: From Passwords to Public Keys*, Addison-Wesley, 2001.

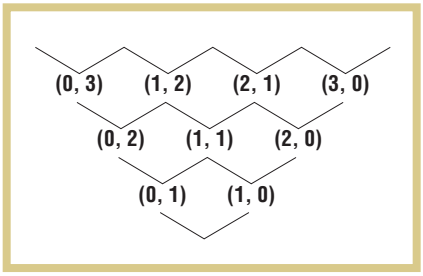

Figure 4. Trust ordering for an e-purse.

It returns the trust calculation and risk assessment to the request analyzer, which can then provide a decision to *p* regarding possible interaction.

Applications

Given the project's exploratory nature, it's important to evaluate the proposed mechanisms in the context of real applications. Here are two applications on which the project is working. Each application deals with numerous interacting entities. The entities might be strangers, and we can't rely on the presence of a centralized service for security. Although the two applications are very different, we are developing instantiations of the framework presented earlier that will be used to support both of them.

Electronic purse

An electronic purse, or *e-purse*, is a device that stores money in electronic form and enables the transfer of money units to other e-purses. Mobile telephones are ideal candidates for implementing e-purses—they have smart cards for securely storing sums of money and are widespread enough for an e-purse system to be deployed. In this scenario, you can transfer funds from the e-purse to the bank and vice versa via a GSM (Global System for Mobile Communications) or Internet connection; you make bank payments offline using SMS (Short Message Service) or Bluetooth.

An e-purse's main advantage over a traditional purse is that a person doesn't need to carry real money. When a real purse is lost or stolen, the money is also lost or stolen; with an e-purse, a thief must know the e-purse's PIN to access its contents. Also, an e-purse should maintain the anonymity property of payments—the buyer can pay the vendor without revealing his or her identity. A payment protocol that is particularly suited to the e-purse is the *eCash* protocol from DigiCash.[16,17] In this protocol, each electronic coin has a unique serial number and the bank signs it using a key for that denomination. When a vendor who receives money cashes the e-purse contents with the bank, the bank detects cheating by comparing these coins and their serial numbers to those already cashed.

A principal in the e-payment system is a purse. While an e-purse typically is owned by a person, it might also be a payment device in a coffee or chocolate vendor. You recognize the principal through its public key, which is stored on its e-purse, *PubPurse*. The Entity Recognition subsystem ensures that the principal is what it claims to be. Both buyer and vendor principals receive a certificate containing their respective public key *PubPurse* signed by the bank: $\{PubPurse, N\}_{\text{Bank}}$. *N* denotes the e-purse's serial number, which is used for principal identity information.

You can verify any request using the *PubPurse* of the initiator and its certificate. Because payment happens offline, the bank can't be contacted to check the certificate's validity and the principal risks dealing with a principal that the bank recently blacklisted. To reduce the risk from replay attacks, where an attacker replays messages from other principals that it has overheard, *PubPurse* can be used in a challenge-response protocol relying on past interactions to increase confidence in the recognition level. For example, the ticket vending machine sends a request encrypted with the buyer's *PubPurse* asking how many tickets have been bought so far and the buyer should reply with the right answer encrypted with the vending machine's *PubPurse*.

In this system, the vendor who accepts e-cash for goods has the biggest risk. The vendor must wait until he or she goes online to deposit the cash to verify that it wasn't forged. There are thus two outcomes for a payment transaction: It succeeds if the bank accepts the money the vendor received, or it fails.

The first task of this application scenario's security administrator is to propose trust values and an ordering for them. Because you can link trust in a principal to the number of positive experiences—the number of successful payment outcomes compared to the number of outcomes where the principal cheated—you can represent trust by a pair (m, n) of non-negative integers. Integer *m* represents the number of successful outcomes associated with the principal; *n* represents the number of unsuccessful outcomes.

The bottom, or unknown value is (0, 0). A trust value (m_1, n_1) represents more trust than (m_2, n_2) when the number of successful outcomes is superior and the number of unsuccessful outcomes is inferior ($m_1 \geq m_2$ and $n_1 \leq n_2$). The information ordering on the set of values is defined by considering a trust value (m_1, n_1) as conveying more information than a value (m_2, n_2) if it's possible to start from the value (m_1, n_1) and then perform some additional number of interactions, ending up with the value (m_2, n_2). This intuition leads us to define the information ordering by taking $(m_1, n_1) \sqsubseteq (m_2, n_2)$ if and only if $(m_1 \leq m_2)$ and $(n_1 \leq n_2)$. Figure 4 illustrates this partial ordering of trust values.

A principal's trust in another is mainly based on its experience with that principal. It can also be influenced by observations of that principal. In the e-purse system, observations take the form of messages ("principal *p* cheated" or "principal *p* is honest") exchanged via princi-

pals, perhaps as an annex to payment messages. The security administrator can write the system's trust management policy to update the trust policy based on these messages. Obviously, such a message from the bank is more credible than one from another principal.

The second task for this system's security administrator is to design the *risk policy*. In this scenario, we deal with two mutually exclusive outcomes. The cost associated with the payment action is bounded by the amount of money being exchanged. One risk policy is to consider the probability of success to be independent of the cost (for example, an attacker is just as likely to cheat for 5 euro as he is for 50 euro). In this case, the cost-PDF is flat. Another possible risk policy is to consider the probability of cheating to be proportional to the amount being transferred. In this case, the probability of an unsuccessful outcome for an amount s of money could be defined as: $p \times a \times s^2$, where p is the trust parameter for the paying principal, calculated as $n/(n + m)$ for the trust values except for bottom (where $n + m = 0$), and a is a constant defined to scale the risk value calculated to the range [0..1]. This cost-PDF models a scenario in which the risk of cheating increases to the square of the sum of money involved in the transaction. A third PDF shown in Figure 5 is a linear curve defined as $amount \times T$, where T is based on the average degree of cheating that occurs in the e-payment system. The risk policy returns this function when the trust value calculated for a principal is bottom (0, 0).

The security administrator's final task is to define the security policy. This policy considers the calculated risk and trust values and decides whether to permit the action. In this system, one possible policy is to print a warning message to the user's screen if the calculated risk value for the transaction exceeds a specified threshold. For example, using pseudocode,

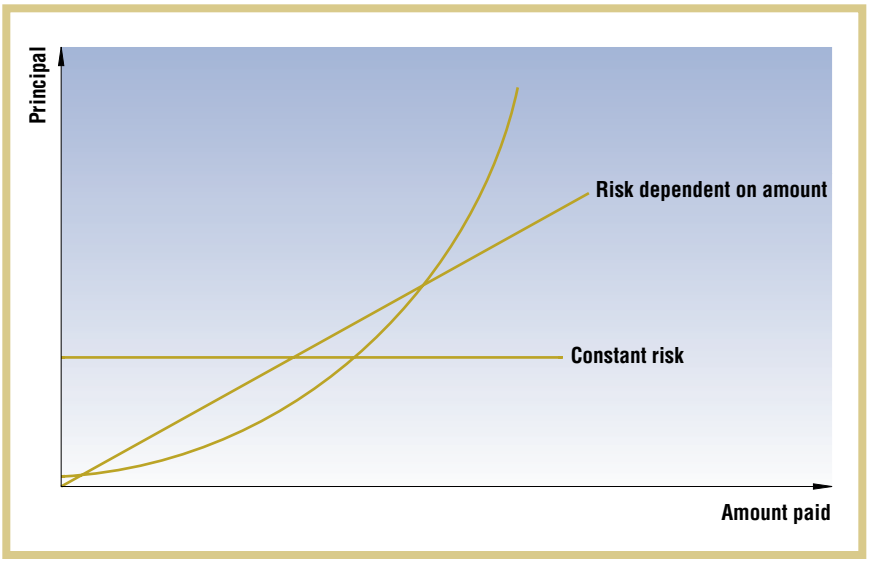

Figure 5. Risk PDFs for the e-purse application.

```
if (riskOfFailurePDF(amount)
    < ThresholdPay)
  return YES;
if (riskOfFailurePDF(amount)
    > ThresholdDontPay)
  return NO;
else
  return DONT_KNOW
```

The riskOfFailurePDF is the PDF returned by the risk policy and amount is the sum being paid in the transaction; ThresholdPay and ThresholdDontPay are the thresholds below and above that the e-purse pays and doesn't pay, respectively. Between these values, the user must intervene in the decision.

For example, Carl might try to buy chocolate from a vending machine. The vending machine trusts Carl, whose trust value is (13, 7). Assuming proportional risk, with $a = 0.2$, ThresholdPay = 0.1, and ThresholdDontPay = 0.2, the answer would be YES for Carl to purchase 1 euro worth of chocolate but NO for 2 euro worth of chocolate.

Collaborative gaming

The demand is increasing for applications such as collaborative gaming, where players in different locations can participate in the same gaming session using portable devices. When money is at stake in these games, security measures are necessary because unknown and potentially untrustworthy players might enter gaming sessions.

Blackjack is a popular card game in which players gamble with a dealer over the value of a hand. In our prototype implementation, people play blackjack over a mobile ad hoc network using laptop computers or PDAs.

For example, suppose Alice takes the 8 a.m. commuter train into the city for work every weekday. She wants to play an interactive game to pass the time, so she joins an ad hoc wireless network to see what collaborative gaming applications are available. She discovers an ongoing blackjack session in which Bob is the dealer and she requests admission to the game. Bob must decide whether or not to admit her; he must decide if he recognizes Alice, how much he trusts her as a gaming opponent, and how much risk is associated in playing blackjack with her.

The entity recognition process determines whether or not Bob has interacted with Alice before, as well as the level of confidence in recognizing her. If this is Bob's first time interacting with Alice, he might need to rely on trusted third-party recommendations about Alice. Recommendations can be exchanged verbally, by email, by distributed post-its, and more, and then recorded as evidence.

On the basis of evidence from his own observations, from recommendations, or both, Bob can determine a trust level for Alice. Bob needs to trust that Alice won't cheat, spoof, or collude while he is gaming with her. This same trust is

required in the casino version of blackjack. Also, because the dealer's odds of winning are more favorable than the players' odds, the players must consider the entity in the dealer role trustworthy. So the right to assume the advantageous dealer role can be considered a privilege earned through fair, trustworthy game playing. Alice must prove some level of trustworthiness if she wants to enter a game as a dealer.

As with e-purse, trust values and an ordering for them are necessary. In blackjack, trust is based on factors such as Alice's typical playing strategy and whether she pays her gambling debts. The security administrator generates an ordering similar to that in the e-purse application that represents both positive and negative experience results.

This information could also help determine the risk of interacting with Alice. For example, information about what players are typically playing in the same games as Alice might enable Bob to reason about the probability that Alice will collude with other players. Bob can monitor this information throughout the course of the interaction and evaluate and store other players' results of playing blackjack with Alice. With his stored information, Bob might also need to respond to requests for recommendations from other players.

Bob might want to delegate his trusting decision altogether. In this case, he must begin recognition of remote entities he could interact with for the purpose of delegation.

Based on this information collection, Bob can assess whether he recognizes Alice, to what extent he is certain of recognition, how much he trusts Alice, and if that trust level is enough to interact with Alice given the overall risk inherent in the interaction.

Initial results of testing the prototype implementation[18] show that it reacts correctly to changes in an entity's interactive behavior—it adjusts trust levels and implements trust-based interaction accurately as trust rises and falls.

We are continuing to refine the framework's design and, in particular, our approach to trust formation and evolution to address issues such as collusion and framing. We are also examining the application of trust to role-based access control. P

ACKNOWLEDGMENTS

The work we describe in this article was partly supported by the Future and Emerging Technologies program of the Commission of the European Union under research contract IST- 2001-32486 (SECURE—Secure Environments for Collaboration among Ubiquitous Roaming Entities).

REFERENCES

1. D.H. McKnight and N.L. Chervany, *The Meanings of Trust*, working paper 96-04, Management Information Systems Research Center, Univ. Minnesota, 1996.
2. D. Gambetta, ed., *Trust: Making and Breaking Cooperative Relations*, 2000, www.sociology.ox.ac.uk/papers/trustbook.html.
3. M. Deutsch, "Cooperation and Trust: Some Theoretical Notes," *Nebraska Symp. Motivation*, M.R. Jones, ed., Nebraska Univ. Press, 1962, pp. 275–319.
4. S. Marsh, *Formalising Trust as a Computational Concept*, doctoral thesis, Dept. Computer Science and Math., Univ. Stirling, 1994.
5. B. Shand, N. Dimmock, and J. Bacon, "Trust for Ubiquitous, Transparent Collaboration," *Proc. 1st IEEE Int'l Conf. Pervasive Computing and Comm.* (PerCom 03), IEEE CS Press, 2003, pp. 153–160.
6. A. Abdul-Rahman and S. Hailes, "Supporting Trust in Virtual Communities," *Proc. Hawaii Int'l Conf. System Sciences* (HICSS-33), IEEE CS Press, 2000, pp. 1769–1777.
7. L. Xiong and L. Liu, "Building Trust in Decentralized Peer-to-Peer Electronic Communities," *Proc. 5th Int'l Conf. Electronic Commerce Research* (ICECR-5), 2002.
8. "The KeyNote Trust-Management System Version 2," tech. memo, Internet Soc., 1999.
9. M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized Trust Management," *Proc. 1996 IEEE Symp. Security and Privacy*, IEEE CS Press, 1996, pp. 164–173.
10. "SPKI Certificate Theory," tech. memo, Internet Soc., 1999.
11. R.L. Rivest and B. Lampson, "SDSI - A Simple Distributed Security Infrastructure," Oct. 1996, http://theory.lcs.mit.edu/~rivest/sdsi10.html.
12. C.M. Jonker and J. Treur, "Formal Analysis of Models for the Dynamics of Trust Based on Experiences," *Proc. Modelling Autonomous Agents in a Multi-Agent World* (MAAMAW 1999), LNAI 1,647, Springer-Verlag, 1999, pp. 221–232.
13. C. English et al., "Trusting Collaboration in Global Computing Systems," *Trust Management*, LNCS 2,692, Springer-Verlag, 2003, pp. 136–149.
14. S. Buchegger and J.Y. Le Boudec, "The Effect of Rumor Spreading in Reputation Systems for Mobile Ad-hoc Networks," *Proc. Workshop on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks* (WiOpt 03), Mobile Information and Communication Systems, 2003; www.mics.org/getDoc.php?docid=341&docnum=1.
15. K. Aberer and Z. Despotovic, "Managing Trust in a Peer-2-Peer Information System," *Proc. 10th Int'l Conf. Information and Knowledge Management* (CIKM-01), ACM Press, 2001, pp. 310–317.
16. D. Chaum, "Achieving Electronic Privacy," *Scientific American*, vol. 267, no. 2, Aug. 1992, pp. 96–101.
17. B. Schoenmakers, "Security Aspects of the Ecash TM Payment System," *State of the Art in Applied Cryptography: Course on Computer Security and Industrial Cryptography*, LNCS 1,528, Springer-Verlag, 1998, pp. 338–353.
18. E. Gray et al., *Towards a Framework for Assessing Trust-Based Admission Control in Collaborative Ad Hoc Applications*, tech. report 66, Dept. Computer Science, Trinity College Dublin, 2002.

Vinny Cahill is an associate professor of computer science and leads the Distributed Systems Research Group at Trinity College Dublin. His research interests address middleware for mobile and sentient computing. He received his PhD from Trinity College Dublin. Contact him at the Distributed Systems Group, Dept. of Computer Science, Trinity College, Dublin 2, Ireland; Vinny.Cahill@cs.tcd.ie.

Elizabeth Gray is a PhD student in the Distributed Systems Group of the Department of Computer Science at Trinity College Dublin. Her research interests include trust-based security measures for ubiquitous computing. She received her BS from Georgetown University and Trinity College Dublin. Contact her at the Distributed Systems Group, Dept. of Computer Science, Trinity College, Dublin 2, Ireland; Liz.Gray@cs.tcd.ie.

Jean-Marc Seigneur is a PhD student in the Distributed Systems Group of the Department of Computer Science at Trinity College Dublin. His research interests include ubiquitous computing security, entity recognition, and technical trust. He received his French engineering diploma in electrical engineering from the French National Institute of Applied Sciences and his MSc in networks and distributed systems from Trinity College Dublin. Contact him at the Distributed Systems Group, Dept. of Computer Science, Trinity College, Dublin 2, Ireland; Jean-Marc.Seigneur@cs.tcd.ie.

Christian D. Jensen is a lecturer in the Department of Computer Science at Trinity College Dublin, where he is affiliated with the Distributed Systems Group. His research interests include computer and network security, especially issues involving secure collaboration among mutually suspicious principals, and relaxed trust models in ad hoc computing. He received his MSc in computer science from the University of Copenhagen and his PhD in computer science from Universite Joseph Fourier in Grenoble, France. Contact him at Informatics & Mathematical Modelling, Technical University of Denmark, DK-2800 Lyngby, Denmark; Christian.Jensen@imm.dtu.dk.

Yong Chen is a PhD student in the Distributed Systems Group of the Department of Computer Science at Trinity College Dublin. His research interests include risk analysis, data mining, and pattern recognition in ubiquitous computing. He received his MSc in image processing and pattern recognition from the School of Electronics and Electric Engineering at Shang Hai JiaoTong University. Contact him at the Distributed Systems Group, Dept. of Computer Science, Trinity College, Dublin 2, Ireland; Yong.Chen@cs.tcd.ie.

Brian Shand is a PhD student in the Opera research group at the University of Cambridge Computer Laboratory. His interests include trust modelling and computerized resource contracts in distributed systems. He received his MSc (Eng) and his BSc in electrical engineering from the University of Cape Town. Contact him at the Univ. of Cambridge Computer Laboratory, JJ Thomson Avenue, Cambridge CB3 0FD, UK; Brian.Shand@cl.cam.ac.uk.

Nathan Dimmock is a PhD student at the University of Cambridge Computer Laboratory. His research interests include trust-management systems, security and privacy in ubiquitous computing, and middleware. He received his BA in computer science from the University of Cambridge. He is a member of the IEEE and British Computer Societies. Contact him at the Univ. of Cambridge Computer Laboratory, JJ Thomson Ave., Cambridge CB3 0FD, UK; Nathan.Dimmock@cl.cam.ac.uk.

Andrew Twigg is a PhD student at the University of Cambridge Computer Laboratory. His research interests are Internet-scale computation and reputation systems. He received his BS in computer science at Warwick University. He is a member of the IEEE. Contact him at the Univ. of Cambridge Computer Laboratory, JJ Thomson Ave., Cambridge CB3 0FD, UK; Andrew.Twigg@cl.cam.ac.uk.

Jean Bacon is a reader in Distributed Systems at the University of Cambridge Computer Laboratory. She is an IEEE Senior Member and EIC of Distributed Systems Online. Contact her at the Univ. of Cambridge, Computer Laboratory, JJ Thomson Ave., Cambridge CB3 0FD, UK; jmb@cl.cam.ac.uk.

Colin English is a PhD student in the Department of Computer and Information Sciences at the University of Strathclyde. He received his BEng in chemical engineering and MSc in information technology systems from the University of Strathclyde. Contact him at the Univ. of Strathclyde, Dept. of Computer and Information Science, Livingstone Tower, 26, Richmond Street, G1 1XH Glasgow, Scotland; Colin.English@cis.strath.ac.uk.

Waleed Wagealla is a research fellow in the Department of Computer and Information Sciences at the University of Strathclyde. He received his BSc in computer science from the University of Khartoum, Sudan and his PhD in computer science from Nottingham Trent University. Contact him at the Univ. of Strathclyde, Dept. of Computer and Information Science, Livingstone Tower, 26, Richmond Street, G1 1XH Glasgow, Scotland; Waleed.Wagealla@cis.strath.ac.uk.

Sotirios Terzis is a lecturer in the Department of Computer and Information Sciences at the University of Strathclyde and a PhD candidate in the Computer Science Department at Trinity College Dublin. His research interests include context-awareness and trust management in pervasive computing systems. He received his BSc and MSc with a specialization in distributed systems from the University of Crete. He is a member of the ACM, the IEEE Computer Society, and the British Computer Society. Contact him at the Univ. of Strathclyde, Dept. of Computer and Information Science, Livingstone Tower, 26, Richmond Street, G1 1XH Glasgow, Scotland; Sotirios.Terzis@cis.strath.ac.uk.

Paddy Nixon is professor of computer science at the University of Strathclyde where he leads the Global and Pervasive Computing Group. He received his BS and PhD in computer science from Liverpool University and his MA from Trinity College Dublin. Contact him at the Univ. of Strathclyde, Dept. of Computer and Information Science, Livingstone Tower, 26, Richmond Street, G1 1XH Glasgow, Scotland; paddy@cis.strath.ac.uk.

Giovanna di Marzo Serugendo is a lecturer at the University of Geneva. Her research interests are formal specifications and verification applied to mobile agents, engineering of self-organizing applications, and trust-based security. She received her PhD in software engineering from the Swiss Federal Institute of Technology in Lausanne (EFPL). She is member of the ACM. Contact her at the Univ. of Geneva, Centre Universitaire d'Informatique, 24, rue Général-Dufour, CH-1211 Genève 4, Switzerland; Giovanna.Dimarzo@cui.unige.ch.

Ciarán Bryce is an assistant professor at the University of Geneva. His research interests include computer security, object orientation, and wireless information systems. He received his PhD from the University of Rennes. Contact him at the Univ. of Geneva, Centre Universitaire d'Informatique, 24, rue Général-Dufour, CH-1211 Genève 4, Switzerland; Ciaran.Bryce@cui.unige.ch.

Marco Carbone is a PhD student in computer science in Basic Research in Computer Science (BRICS) at the University of Aarhus. His research interests are concurrency theory, static analysis, semantics of computation, and security and models for trust-based systems. He received his MSc in computer science from the University of Pisa. Contact him at BRICS, Dept. of Computer Science, Univ. of Aarhus, Ny Munkegade, building 540, 8000 Aarhus C, Denmark; carbonem@brics.dk.

Karl Krukow is a PhD student in computer science in BRICS at the University of Aarhus. His research interests are semantics, programming languages, security, applications of category theory, and foundations for trust-based systems. He received his BA in computer science from the University of Aarhus. Contact him at BRICS, Dept. of Computer Science, Univ. of Aarhus, Ny Munkegade, building 540, 8000 Aarhus C, Denmark; krukow@brics.dk.

Mogens Nielsen is an associate professor and director of the BRICS International PhD School at the University of Aarhus. His research interests include process algebras, models for distributed computations, concurrency, and logics in computer science. He received his PhD in computer science from the University of Aarhus. Contact him at BRICS, Dept. of Computer Science, Univ. of Aarhus, Ny Munkegade, building 540, 8000 Aarhus C, Denmark; mn@brics.dk.