

# $\mathcal{X}$ Trust: A Scalable Trust Management Infrastructure

Marc Branchaud  
 RSA Security, Inc.  
 Vancouver, BC, Canada  
 marc narc@rsasecurity.com

Scott Flinn  
 National Research Council of Canada  
 Institute for Information Technology  
 Scott.Flinn@nrc.gc.ca

**Abstract**—This paper introduces  $\mathcal{X}$ Trust, a general purpose security framework that supports a wide range of applications and security mechanisms in a way that makes trust highly visible. It provides basic services for identifying paths between peers that are trusted for specific purposes, and for utilizing those paths for the trustworthy exchange of information. The design of the system is based on four general principles: (1) trust is always dependent on context; (2) trust relationships are defined by people, not applications, so trust should be dealt with separately from applications; (3) trust relationships are defined using local semantics; and (4) trust, when it exists, is absolute in that information from a trusted party is accepted without question.

## I. INTRODUCTION

Many of the digital communication technologies on which our society now depends were conceived at a time and in an environment where security, privacy and trustworthiness were not significant concerns. Since that time, there has been an ongoing qualitative change in the way the Internet is used. It is now a platform for electronic commerce, an engine for streamlining business processes, a vehicle for seeking out and maintaining interpersonal relationships, a distribution channel for copyrighted works, and a forum for social discourse. As a result, the sensitivity of the information the Internet carries has far surpassed the protections provided by the protocols upon which it is built.

These shortcomings have generally been addressed in one of two ways. First, protocols such as IPsec and TLS augment existing network services, providing basic encryption and authentication services to applications. However, the trust relationships supported by these protocols lack semantic context. This makes it difficult to leverage a security infrastructure established for one protocol beyond its initial application. For example, the public key infrastructure (PKI) that supports secure web shopping has not been leveraged to support federated identity management or secure instant messaging. This despite the fact that the standards upon which the infrastructure was built were designed for general purposes and are application agnostic.

The second approach is to integrate trust relationships and semantics at the application level. For example, reputation systems, as typified by the eBay auction site, facilitate commercial transactions between parties having no prior knowledge of each other. Social networks have attracted considerable attention, ranging in purpose from simple “friend finders” such as

Friendster to more sophisticated community-of-interest applications like Yenta [5] and Acorn [13]. Generally speaking, security, privacy and trust assurances in these applications are provided by the applications themselves. As a result, the reputation systems and communication networks they provide tend to be closed and not easily leveraged for other purposes.

Despite much work on these two approaches to security, it is still true today that new and emerging applications do not have a widespread, general-purpose security system they can rely upon, and so most new applications opt to re-invent their security, or to ignore security altogether.

In this paper we describe a system we call  $\mathcal{X}$ Trust (pronounced *ex-trust*) that combines the strengths of these two approaches to provide a general purpose security framework for trust management. In this context, and throughout the paper, we use the term *trust* in the sense suggested by the following definition from Jøsang and Presti [19]:

*Trust is the extent to which one party is willing to depend on somebody, or something, in a given situation with a feeling of relative security, even though negative consequences are possible.*

The design philosophy of the  $\mathcal{X}$ Trust system is based on four central ideas:

- **Trust depends on context.** For example, a party who trusts another for the purpose of providing the correct public key for an identity may not also trust that same party to recommend a good restaurant. Trust relationships between entities are meaningful only for specific contexts and should not be used to infer trust outside of those contexts.
- **Trust relationships are defined by people, not applications.** Trust should therefore be abstracted from the applications that leverage it. A network of trust relationships that explicitly represents the context in which each relationship applies can be used by any application to find paths to entities that are trusted for specific purposes.
- **Trust relationships are defined using local semantics.** Trust is a complex social phenomenon that is not easily captured with a single set of labels and numbers. The concepts involved in the semantics of a specific trust relationship – the terms and conditions governing its use – depend not only on the trust context, but on factors such as the culture of the people and organizations involved, the political and regulatory environment, and organiza-

tional boundaries and intellectual property concerns. Even within a specific trust context, these factors may vary by geographic region, political jurisdiction, and so on. We believe that it is unreasonable to depend on achieving widespread agreement on such things. A general purpose trust network should therefore be able to define trust relationships using terminology and semantics that may be unique to the partners involved.

- **Trust is absolute.** As suggested by the definition quoted above, trust is at play in exactly those situations where you must rely on someone or something to act in your best interest, even though you are unable to constrain them to do so. Although you are always free to act upon information in any way you like, when you ask a trusted partner a question you cannot constrain how your partner obtains the answer because you have no way to verify that the constraints were observed. You trust your partner to act appropriately. If you ask a trusted authority for the public key corresponding to a given identity you must accept the key returned to you as the authority's own reply, even if that authority relies on its own trusted intermediary who may have an opportunity to deliver the wrong key to facilitate a man-in-the-middle attack on data encrypted with it. When you ask friends for opinions about the trustworthiness of a Web site, if you choose to rely on those opinions then you must trust your friends to reach their conclusions in sensible ways and to return answers that accurately and honestly reflect their beliefs. If you ask a partner a question using a representation that is not universally recognized, you must trust them to maintain the fidelity of the question through the semantic mapping process, even declining the request if necessary. They in turn must trust any partner to whom they refer the question to do the same.

$x$ Trust defines a *trust network*. Communication in the  $x$ Trust system is routed through a network of  $x$ Trust entities in a manner that is analogous to the routing of IP packets through the Internet. First and foremost,  $x$ Trust is a system for identifying paths between end points and realizing the secure exchange of information through those paths.

Building on this foundation, the  $x$ Trust system provides a generic platform for trustworthy communication, and the four ideas outlined above represent its design goals. It facilitates communication between trusted parties that is encrypted and mutually authenticated by default, combining the four ideas in a way that gives it a number of relative strengths in comparison with existing models and frameworks (see Section IV for a more detailed discussion). Specifically:

- **$x$ Trust facilitates context selection.**  $x$ Trust in fact defines a *set* of overlapping networks, each one unique to a specific trust context. The idea of *context selection* is central to  $x$ Trust. The selection of a particular context determines the subset of the full trust network that will be utilized in satisfying a request.
- **$x$ Trust is application neutral.** It provides a small number

of basic services that can be used by applications to achieve a wide range of goals. Section II describes the services, and Section III contains examples of how the services may be composed to support familiar applications. The ability of  $x$ Trust to deal with different trust contexts allows it to provide application-specific trust services without having to build assumptions about context into the network. For example,  $x$ Trust is equally well suited to federated authentication and secure messaging. Other approaches may be good at one or the other, but they offer little opportunity to leverage the trust investment in each technology for other purposes.

- **$x$ Trust scales well.**  $x$ Trust explicitly avoids any assumption of a small number of global authorities. Such assumptions significantly limit, and in our opinion weaken, the trust assurances a system can provide. Furthermore,  $x$ Trust does not rely on a closed-world assumption and can therefore build networks between entities who do not all agree on common semantics. This makes it easier to add entities incrementally because new entities can operate with semantic variations that have no impact on the semantics in use elsewhere in the network. Current security infrastructures suffer from one or both of these assumptions, either in their design or implementation (see Section IV for more).
- **$x$ Trust makes trust visible.** All security systems require users to make trust decisions. A system that appears to eliminate this need is really only hiding it, which is worse. As you read this paper, it may strike you that dependence on absolute trust creates many opportunities for betrayal. If so, then you will have recognized (and demonstrated) one of the relative advantages of  $x$ Trust over alternative schemes: users deploying an  $x$ Trust system will be highly aware of their trust vulnerabilities.

These ideas are all reflected to some degree in existing technologies for secure communication. However, we believe the combination of trust context selection based on local semantics with the idea of using routing for trust path discovery in an application-agnostic trust network to be completely unique. Parties interacting in cyberspace trust each other to widely differing degrees and for widely varying purposes, just as they do when interacting offline. It is the particular combination of ideas embodied by  $x$ Trust that allows it to excel at accommodating this diversity.

## II. THE $x$ TRUST SYSTEM

$x$ Trust is a peer-to-peer trust network of *Domains* connected through *trust links*.  $x$ Trust *Principals* – people, computers, organizations, etc. – that wish to use the system must belong to at least one Domain. A Domain can represent one or more Principals. In practice, it is helpful to think of each  $x$ Trust Principal as having its own private Domain which the Principal uses to manage its own trust.

Trust links between Domains are one-way: Domain A might trust Domain B, but B might not trust A. If A trusts B, then A is called the *source* of the trust link between A and B, and

B is called its *sink*. Graphically, a trust link is represented as an arrow from source to sink. Figure 1 shows a simple trust network.

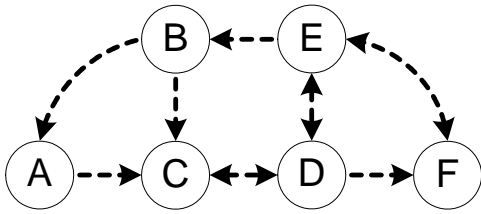


Fig. 1. Simple trust network. To reduce clutter, double-headed arrows represent two trust links.

A trust link is established by direct negotiation between the source and the sink. This negotiation occurs outside the  $x$ Trust system:  $x$ Trust itself does not provide any mechanism for establishing new trust links.

The set of trust sinks that are directly linked to a single source are the source's *trusted peers*. The  $x$ Trust system allows a Domain to leverage its trusted peers to make determinations about any Principal in the network. Domains use a simple request-response protocol to communicate with their trusted peers and ask them questions about Principals. All communication in  $x$ Trust occurs between peers directly connected with trust links.

An  $x$ Trust question about a Principal can take two forms. The first seeks to obtain a value for some attribute of the Principal. For example, what is a Principal's e-mail signature verification public key? These attribute questions have only a single answer, and the answer is either right or wrong. The second form of question aims to establish some kind of opinion about the Principal. For example, can the Principal be trusted to keep credit-card data confidential? These reputation questions have no single answer, nor is any given answer absolutely right or wrong.  $x$ Trust is a single system that can provide answers to both kinds of questions.

One Domain will trust another in a given context for a particular set of attributes and opinions. We use the phrase *trust predicates* to refer to this set. The source of each trust link has a set of trust predicates that enumerate what information the sink is trusted to provide.

#### A. Context

*Context* is a fundamental aspect of the  $x$ Trust system. Trust links are context-sensitive: a link from Domain A to Domain B might exist in some situations but not others. When the link exists in several contexts, Domain A's trust predicates can also be different in each context. The source of the trust link, Domain A, determines the contexts in which the link exists and which predicates apply in each context. Since a Domain's set of trusted peers changes depending on context, context affects the entire trust network. The example in Figure 1 in fact only shows the trust network for one context. A different context would entail a different set of Domains and a different arrangement of trust links.

A context can be any application or situation that requires an  $x$ Trust Domain to make decisions. When a trust link is established, the source Domain identifies which of its contexts can use the link. Each context represents a different set of trusted peers and trust predicates. For example, a Domain might have a `casual-email` context that specifies a large set of trusted peers and some broadly applicable predicates. The Domain may also have a `project-x` context where the trusted peers are limited to the partners involved in Project X, and the predicates are highly specialized. A context may specify other parameters such as a time-of-day restriction, requestor authentication, or any other factor that the Domain feels has a bearing on its decisions.

Every Domain operates within its own set of contexts, and each of a Domain's contexts has its own local name. Every  $x$ Trust request is associated with a context that the Domain uses to decide how to respond.

#### B. $x$ Trust Identifiers

Every Principal in a Domain is assigned an identifier called an  $x$ ID. An  $x$ ID consists of two parts:

- 1) A *Domain Identifier* (DI) that is globally unique and allows entities to recognize the  $x$ IDs of Principals belonging to a particular Domain.
- 2) A *Principal Identifier* (PI) that is a locally unique name for a Principal within the Domain.

It is syntactically convenient to represent an  $x$ ID as the PI, followed by an @ sign, followed by the DI. For example, `jonas@example.org` is an  $x$ ID where the Principal Identifier is `jonas` and the Domain Identifier is `example.org`. The  $x$ Trust system places no restrictions on the format and contents of the parts of an  $x$ ID: any string can be used as a Domain or Principal Identifier. What's important is that every  $x$ ID consists of these two parts.

Furthermore, there are no restrictions on how many  $x$ IDs a Principal may have in any given Domain (a Principal is said to *own* an  $x$ ID). A Principal may also belong to more than one Domain. This allows a single real-world Principal to have multiple  $x$ IDs, which it can selectively use in different situations. Since  $x$ Trust provides no way to link the different  $x$ IDs, the Principal can act pseudonymously through its multiple identities.

$x$ IDs provide unique identifiers with which to associate attribute values and opinions. An  $x$ ID's Domain Identifier provides a nominal way to recognize when a Principal belongs to one's own Domain, a trusted peer Domain, or an unknown Domain. However, the DI does not necessarily identify a Domain that is authoritative about the Principal in some particular way. In  $x$ Trust, authorities are locally defined in each Domain, and shift according to context.

Some see a requirement for globally unique names as cause for consternation. We note that these concerns are mostly theoretical. In practice, globally unique names have turned out to be fairly common in systems like the Internet – it is no accident that  $x$ IDs resemble e-mail addresses or instant-messaging IDs.  $x$ Trust itself does not provide or assume any method

for selecting globally unique Domain Identifiers because such methods are more appropriately defined by the environment in which  $x$ Trust is deployed. For example, an Internet-wide  $x$ Trust system would do well to use Internet domain names as  $x$ ID Domain Identifiers. Many security proposals have been tripped up by defining their own solutions to global naming.  $x$ Trust avoids that trap by making the uniqueness requirement explicit, and by relying solely on the uniqueness of DIs while not relying on any aspect of their content, syntax or semantics. Applications can also avoid mis-identifications due to non-unique DIs by using an active, challenge-response protocol to authenticate Principals that relies on credentials obtained through  $x$ Trust. This will detect DI collisions as another form of impersonation attack.

### C. The $x$ Trust Protocol

The  $x$ Trust protocol is a stateless request-response protocol. Requests ask for an attribute of or an opinion about a *subject* Principal. An  $x$ Trust request also specifies the context in which the request is being made.

The  $x$ Trust protocol takes place between two Domains, one the requestor and the other the responder. The requestor is always the source of a trust link, and the responder is always one of the requestor's trusted peers. Peers can authenticate each other using credentials exchanged when the trust link is established. Requestors always authenticate responders; responders may optionally authenticate requestors.

It is important to recall that trust is absolute. In  $x$ Trust this means that when a responder receives a query it is free to construct its response in any way it sees fit, including consulting other responders.  $x$ Trust does not provide any means for the requestor to have insight into the responder's processes. That said, there is nothing preventing such insight from being communicated by some other means. The upshot is that when a requestor decides to query a responder, it is placing complete trust in that responder to obtain an answer to the question.

Requestors can query responders in two ways. The first is to ask for an authoritative response. This kind of query is used when the requestor believes there is a single source for the requested information. For example, a query for `jonas@example.org`'s public key is a request for authoritative information – the key either exists or it does not, and the response will either contain it or won't. From the requestor's point of view, there is only one authoritative source for the public key.

A requestor sends an authoritative query to a single trusted peer – the one that it believes is most able to provide an authoritative response. If the responder considers itself to be authoritative for the requested information, it simply returns the information to the requestor. If it does not, it relays the request to one of its trusted peers (again, the one it believes is the most likely to provide an authoritative response). This process continues until an authoritative responder is reached. The  $x$ Trust system is in fact routing the request to the authoritative responder, in much the same way that a data network

like the Internet routes packets to a destination. Section II-D discusses more aspects of routing queries in  $x$ Trust.

The second kind of  $x$ Trust query is to ask for an opinion. This uses the  $x$ Trust system as a reputation network. For example, a query for `jonas@example.com`'s trustworthiness as a merchant is a request for reputation information – there is no authoritative source for the information, only various opinions.

A requestor sends an opinion query to one or more, or all, trusted peers – whichever one(s) it thinks can provide a valid opinion. The requestor consolidates the response(s) from its peer(s) in whatever way it sees fit in order to arrive at its own opinion for the query. When a responder receives a reputation request, it responds with its opinion in the request. That opinion may be the result of the responder itself querying its peers and combining their responses in some way, or the responder's opinion (in the given context) may simply be configured into the responder. This process is analogous to multicasting in a data network.

The process of determining the answer to a request, be it authoritative or reputational, is called *resolving* the request.  $x$ Trust adopts the routing techniques used in data networking to resolve requests efficiently.

### D. Trust Routing

A fundamental operation of  $x$ Trust entities is determining which trusted peer(s) to query with a particular request. This determination is very similar to the kinds of decisions that a router needs to make when passing packets around a data network. In  $x$ Trust, responders can be thought of as *trust routers* passing queries around a trust network. A rough analogy can be drawn to the ISO's OSI 7-layer network model, with the Physical and Data Link layers replaced by trust links. Just like layer-3 data routers, trust routers use a routing protocol to help them determine which of their peers is best able to provide an answer to a query.

A trust routing protocol works in much the same way as a data routing protocol. The main differences are:

- With a data routing protocol, peers communicate over physical links, whereas with a trust routing protocol peers communicate over trust links.
- While data routing is concerned with discovering the best (usually the shortest) path to a destination address, trust routing aims to discover the best (usually the most trustworthy) path to an authoritative Domain. The metrics that determine a path's suitability are different, although the underlying algorithms are the same.
- Trust routing for a reputation network employs multicast techniques rather than specific path discovery algorithms.

As with data network routers,  $x$ Trust responders participate in a routing protocol, performing a distributed calculation to determine the optimal peer that can respond to a query. As trust is context-sensitive, a Domain performs these calculations for each of its contexts.

A full investigation of trust routing will have to wait for another paper. At this point, we can only offer a few

preliminary observations. It appears that  $x$ Trust’s need for context sensitivity makes trust routing most closely resemble what the data-routing community terms “policy-based” routing. Policy-based routing has been called “almost intractable” [16]. We believe that the characteristics of the  $x$ Trust system, particularly the independence of each Domain to respond to queries in its own way and the absence of any need for global definitions of contexts or other metrics, make trust routing tractable.

It has also been noted that applying multiple metrics to data routing slows the routers down. Data routers must make a routing decision for every data packet on the network, however a trust router need only make a routing decision for each trust query. We expect the amount of “trust traffic” to be significantly less than the amount of data traffic, enough so that more complex routing decisions can be made by trust routers without degrading performance unacceptably.

### E. Semantics

The  $x$ Trust system allows each Domain to adopt its own semantics for dealing with trust and security issues. Each Domain’s set of contexts and trust predicates is independent of every other Domain’s. This enables an  $x$ Trust network to grow in a scale-free fashion, allowing new Domains to connect to the network without having to reconfigure all existing Domains. (Scale-free networks are known to be susceptible to coordinated attacks [3]. However, a scale-free approach may be the best way to address the diverse trust and semantic requirements of a global security infrastructure. Further work is needed to understand the scale-free properties of  $x$ Trust and their effects.)

Although the syntax of the  $x$ Trust protocol is fixed, the semantics of its payloads – the queries being made, the contexts involved, and the contents of the responses – are subject to local interpretation in each Domain. Inter-Domain payload semantics are negotiated as part of the process of establishing a trust link between the Domains. Thus, each Domain understands how its semantics relate to those of its peers. This enables the Domain to change the semantics of a query as it passes the query to a neighbouring Domain. We call this process *semantic mapping*.

$x$ Trust abstracts semantic mapping out of the core processing of the system, allowing each Domain to fully define how its own semantics relate to other Domains’ semantics. Semantic mapping is invoked whenever a query is received from or sent to another Domain. Each trust link requires its own semantic map. Semantic mapping may be based on well known formalisms or hand-crafted for specific purposes, and is used to translate requests and responses as they flow through the network.

Of course, Domains are free to agree on a common set of semantics, although  $x$ Trust itself imposes no such requirement. We anticipate that a global set of common semantics could arise, covering wide-spread activities such as low-assurance signed e-mail verification or shopping on the web.

In general, we expect semantic mapping to be commonplace. There are several techniques that can be applied to negotiating semantics between Domains. Strictly speaking, these are beyond  $x$ Trust’s scope, although  $x$ Trust itself is an enabler for technologies such as ontologies, rule engines, and XSLT, and perhaps even for the Semantic Web [20].

### F. Example

This section presents an example of how one Principal, `larry@D`, can use  $x$ Trust to obtain a message authentication credential for another Principal, `moe@A`. The trust network is the one shown in Figure 1. For this example, the authentication credential is `moe@A`’s signature-verification public key. We use a public-key cryptography credential because it allows us to avoid complicating the example with a discussion of in-transit credential-protection mechanisms. In general,  $x$ Trust is credential agnostic and can support any kind of credential.

Another simplifying assumption made for this example is that Domain A is authoritative for `moe@A`’s public key. Recall that authoritativeness is relative in  $x$ Trust, so the authority of an attribute ultimately depends on who is asking. This simplification allows us to focus on other aspects of  $x$ Trust.

All the Domains in this example participate in a trust routing protocol so that each one knows the optimal neighbour to which requests should be forwarded. The example begins when Principal `larry@D` submits a query to the  $x$ Trust service running in Domain D. This query specifies the following parameters:

- **Subject Principal:** `moe@A`
- **Context:** Message Authentication
- **Attribute:** Signature-verification-public-key

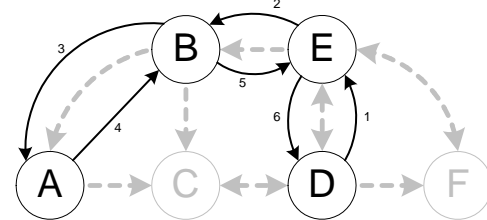


Fig. 2. Sequence of events in resolving an  $x$ Trust request.

Figure 2 shows the ensuing sequence of events:

- 1) Domain D’s  $x$ Trust service consults its trust routing tables and determines that the best neighbour that it can ask for `moe@A`’s public key is Domain E. Domain D maps the request’s semantics into Domain E’s semantics, essentially substituting Domain E’s terms for the Context and Attribute parameters, and sends the new request to Domain E:

- **Subject Principal:** `moe@A`
- **Context:** Public-key-signature-authentication
- **Attribute:** Key-value

Recall that semantic mapping can occur at each hop. For brevity we will no longer mention it in this example.

- 2) Domain E's  $x$ Trust service consults its routing tables and decides to forward the request to Domain B.
- 3) Domain B decides to forward the request to Domain A.
- 4) Domain A, considering itself to be authoritative for moe@A's public key, returns the key in a response to Domain B.
- 5) Domain B returns the key in its response to Domain E.
- 6) Domain E returns the key in its response to Domain D.

Finally, Domain D returns the key to larry@D, who can use the key to authenticate moe@A's message.

This example highlights some important aspects of  $x$ Trust:

- The example shows how  $x$ Trust promotes the use of transitive trust to achieve a large-scale infrastructure. However, the decision to take advantage of transitivity is under the full control of each Domain. This is another area where context plays an important role. In some contexts, a Domain's policies may insist that subject Principals belong to a trusted peer Domain. The Domain Identifier in the Principal's  $x$ ID can help a Domain ensure that it only accepts Principals that belong to a trusted peer Domain. In the above example, if Domain D's policy (in the given context) was such that it would only allow authentication of Principals from trusted peer Domains, then larry@D's request for moe@A's key would fail because Domain A, the authority for the query, is not one of Domain D's trusted peers. In other contexts and with other policies, a Domain may be perfectly willing to have a trusted, non-authoritative Domain forward the request on to some other Domain, as was shown in the example.
- Requests and responses are transformed at each hop to accommodate the receiving Domain's semantics. Similarly, because the queries and responses are routed along trust links that only exist between peer Domains, authentication of the  $x$ Trust messages is also performed at each hop.  $x$ Trust's hop-by-hop authentication leverages transitive trust to enable larry@D to authenticate moe@A across the network.
- The  $x$ Trust system allows for highly dynamic environments, with trust relationships being downgraded or severed at any time. Just as routing algorithms in data networks ensure optimal connectivity, routing algorithms in  $x$ Trust ensure optimal security.
- Note that Domain D has no way to know that Domain E did in fact forward the request to Domain A. This is true for any requestor – the best it can do is to submit the request to a trusted Domain and trust that the request will be processed properly. A malicious or compromised Domain can wreak havoc, but this is true of any trusted entity in any security system, because trust is absolute. *Caveat truster!*
- Response values can be cached by any Domain along a trust path. Caching introduces some interesting dynamics into the  $x$ Trust system, which we mention only briefly here. Cached data has implications for both performance

and security, and there seems to be a trade-off where high-security applications would demand little caching in the system. Requests can include cache-control parameters to notify Domains of the requestor's needs for fresh data. This allows a single  $x$ Trust system to accommodate a spectrum of high- to low-security applications, with each application specifying its caching requirements. Applications that required the freshest data could also be made to pay for any extra facilities needed to meet their performance requirements.

### III. APPLICATIONS

The  $x$ Trust system provides basic services for high assurance exchanges of information with trusted peers. It is up to applications to use them effectively. This section describes how a variety of familiar applications and services can be built using  $x$ Trust.

#### A. Authentication

Suppose two organizations would like to authenticate each other's members. It might be two universities who would like to recognize each other's students, or two publishers who would like their subscribers to be able to access each other's publications. In the simplest case, each party would establish a trust link with the other. In a typical transaction, a member of organization A might request access to the Web site of organization B, asserting her identity (her  $x$ ID) to site B. To authenticate the client, the Web application at site B would ask its local  $x$ Trust responder for the public key for the asserted identity. The responder would obtain the key through the mechanism described in the example of Section II-F above, routing the request to organization A. Trusting that it has received the correct public key for the given identity, site B would then issue a proof of possession challenge (by, for example, requiring client authentication in a TLS handshake) to confirm that the originator of the request controls the corresponding private key, thus completing the authentication.

Authentication of a server is similar with respect to the protocols involved, but the trust relationships are typically quite different. In the cross-Domain client authentication example above, the trust relationships are at the level of organizations, not individuals. It would be impractical for individuals to negotiate formal trust relationships with every organization whose Web site or Internet service they would like to use. Yet when a user visits an organization's Web site, she would like to be assured that she really is dealing with that organization. What point of entry into an  $x$ Trust network might connect her with an arbitrary organization?

This is largely the role that commercial certificate authorities (CAs) play today in the context of Web server authentication. They aim to protect against DNS spoofing and similar threats by providing assurance that the identity of the Web site operator is authentic. In the  $x$ Trust view, the same organizations could fill the same role but the authentication mechanism would be different.

Users would establish trust links with selected *Trust Service Providers* (TSPs) for a server authentication context, mirroring the existing practice of maintaining a store of trusted CA root certificates. When establishing a connection with a server, a client would request the server's public key from a TSP and then issue a proof of possession challenge (through TLS, say) to the server to confirm that it controls the corresponding private key.

There are a number of ways that a TSP could obtain a server's key to respond to the request. For example, a TSP could act as a registrar of server keys for various organizations. The TSP could then respond to public key queries with keys taken from its own local store. Alternatively, TSPs could establish trust links with organizations that operate their own *xTrust* service to publish their own servers' keys. The TSP would satisfy client requests by passing the query to the responsible organization. A standard set of contexts, which may roughly approximate current certificate policies, could be used to avoid the need to separately negotiate the trust predicates for each relationship. Severing a trust link would be equivalent to revoking a certificate, but finer grained adjustments are also possible by restricting or expanding the TSP's trust predicates for each organization.

A hybrid approach is also possible where, if the TSP does not have a direct link to the server's organization and does not have the server's key stored locally, it may refer the request to another TSP. Each approach has strengths and weaknesses with respect to the assurances that can be offered. The flexibility of *xTrust* allows each TSP to choose the arrangement that best meets its own needs. It does not require that all TSPs implement a uniform approach.

Consumers, as relying parties, would enter into contracts with TSPs for reliable trust services. Contrast this with the current practice of trusting public CAs that have no formal relationship with relying parties. Most users today are unaware of the trust relationships with CAs that browser vendors have established on their behalf, and they rarely adjust these relationships. A formal relationship between a user and a TSP would bring the trust decision closer to the individual, creating opportunities for recourse that are not currently available. If a TSP customer learns that her trust has been betrayed, she has the option of transferring her business to another provider. Over time, TSPs would develop reputations for trustworthiness that would create economic incentive for them to continuously monitor and reassess trust relationships.

An attractive approach may be for individuals (or organizations) to look to their Internet Service Provider (ISP) to fill the role of TSP, establishing *xTrust* links directly with their ISP as part of their terms of service with that provider. The ISP would in turn forge links (directly or indirectly) with TSPs that it deems to be trustworthy for various purposes such as server authentication, message authentication (Section III-E), etc. Because its clients will often have the option of seeking service from another provider, trust would become a marketable commodity for the ISP along with other quality of service characteristics.

In both client and server authentication, *xTrust* provides a migration path away from the X.509 PKI without wholesale replacement of the mechanisms that use X.509 certificates. Specifically, *xTrust* can replace the path discovery, validation, and status-verification steps involved in using X.509 certificates. An *xTrust*-aware X.509 application would construct an *xID* from the certificate it wishes to use. The certificate's Issuer Distinguished Name (DN) field would form the *xID*'s Domain Identifier, and the Subject DN field would form the Principal Identifier. Other certificate fields, such as policy identifiers and usage constraints, could serve as input trust predicates to help identify the appropriate *xTrust* context. The application can then simply request the public key for the *xID* and compare the value received to the key in the certificate. An exact match would confirm that the public key in the certificate can be trusted, and the application's X.509-centric protocol could proceed as usual. Adopting this hybrid approach should be fairly painless, as most X.509 toolkits allow certificate validation procedures to be extended or replaced. See Section IV-A for more on *xTrust* and the X.509 PKI.

### B. Authorization

Authorization is the process of determining whether a Principal can access a given resource. Access to a resource is governed by a set of rules that specify the criteria the Principal must meet in order to gain access. These criteria can take many different forms: membership in a group or role; or a set of attributes with minimum values, or a range of values, for example. The criteria that control access to any given resource are determined by the administrators of the resource and are programmed into the access-controlling, or *guardian*, application.

Regardless of the nature of the criteria, authorization fundamentally involves the guardian obtaining some information about a Principal to determine if the authorization criteria are met. The difference between authentication and authorization is that authentication determines which one of the entire population of Principals is on the other side of a communication, while authorization obtains information about that Principal in order to make a security decision.

*xTrust* enables authorization by allowing guardian applications to query the *xTrust* network for information about a Principal. The guardian application is programmed with whatever authorization criteria a resource's administrators deem necessary. When a Principal requests access, the guardian queries the *xTrust* network for attribute values and opinions related to the Principal.

For example, access to a medical database might require that a Principal be a doctor. The database's guardian determines whether a Principal is a doctor by querying the *xTrust* network for an attribute such as *is-a-doctor*. The database administrators configure their guardian with a set of trusted Domains for answering queries about the *is-a-doctor* attribute.

Note how this is different from authorization systems which require the Principal to present their credentials for access

control, usually through a signed assertion or certificate. This approach limits scalability because it requires a Principal to obtain – and understand – authorization credentials for every desired resource. It also requires the guardian to either trust the source of the Principal’s credentials, or for the Principal and guardian to share a trusted source of credentials. This further limits the ability of the system to scale to large populations.

In contrast,  $x$ Trust enables every guardian to have its own set of authorization criteria and trusted sources for those criteria, without requiring any accessing Principal to understand or even be aware of those criteria. Avoiding any need for coordination between guardians and Principals makes  $x$ Trust very scalable.

$x$ Trust’s semantic mapping also enables more flexible authorization. Consider a partnership of publishers, each of whom has a different set of subscription categories, levels and packages. An agreement between publishers A and B might indicate that subscribers of publisher A’s *family* package may also access publisher B’s *home and garden* publications. When a subscriber of A goes to B’s site and requests a gardening publication, B queries the  $x$ Trust network to determine if the subscriber can access its *home and garden* material. When A and B established their  $x$ Trust link, they mapped A’s *family* package to B’s *home and garden* set. So when B queries  $x$ Trust about A’s subscriber, B’s  $x$ Trust Domain automatically asks A’s Domain if the subscriber belongs to A’s *family* package. These mappings can happen at each hop along a path, so even if A and B did not have a direct relationship they could share subscribers as long as their policies, and those of any intermediaries, allow it.

Privacy is an important aspect of a security system. When using  $x$ Trust in this way for authorization, a Principal’s authorization information might pass through Domains unknown to either the Principal or the guardian. This information might be sensitive, and the Principal may not wish it to be widely known. Fortunately,  $x$ Trust allows the Principal to control how information is propagated.

Consider that any Domain that is authoritative about some aspect of a Principal would need to have some kind of relationship with that Principal. If the Principal considers some aspect of himself to be sensitive, he can arrange for the Domain to only divulge that sensitive information to other Domains that he trusts to manage the information appropriately. Thus the Principal has some control over who gets to see his sensitive information. In effect, the Principal could employ *Authorization Service Providers* (ASPs) to publish aspects of himself, in much the same way he could subscribe to TSPs as described in Section III-A. Of course, this does not prevent some other Domain from claiming to be authoritative for this data (which is also true in the offline world). However, the truly authoritative Domain can also keep logs of where queries for that information come from, allowing the Principal some understanding of who is accessing the data. This logging can help resolve disputes that may arise when a false authority makes an erroneous claim about the Principal.

### C. Identity Federation

Identity federation in  $x$ Trust combines the basic authentication and authorization operations (which in turn are both simply basic  $x$ Trust queries with different parameters). Federated authorization follows naturally from the  $x$ Trust authorization process. Federated authentication is achieved through an application provider making the appropriate authentication query.<sup>1</sup>

Federated authentication centers around the logged-in status of a Principal. Application providers can not ask for a Principal’s password directly, so they need an assertion that the Principal has authenticated to the system. In  $x$ Trust, application providers query the  $x$ Trust network for this assertion, for example asking for the value of an `is-logged-in` attribute.

Every application provider relies on its own set of trusted authorities to obtain an authentication assertion. Authentication is only achieved if a trust path exists between the application provider and the Principal’s “home” Domain where the login actually occurred.

The  $x$ Trust approach frees participants in a federated system from having to share a common set of trusted authorities and authentication semantics. When a new application provider joins the system, it selects its own trust anchors, and semantic mappings are established directly and only with those trusted peers. There is no need to coordinate with the rest of the network, or to conform to any common practices or policies. This allows  $x$ Trust to easily scale to accommodate larger and more diverse groups.

Current federated identity systems such as SAML [12] and the Liberty Alliance Project [11] focus on federating authenticated identities, while  $x$ Trust fully supports both authentication and authorization. Furthermore, these systems typically expect participants to share a common set of trusted authorities (“identity providers” in Liberty parlance). Although there are provisions for supporting indirectly-trusted authorities, or relying on a separate infrastructure for authenticating authority assertions, there is no guidance on how indirect trust should work.  $x$ Trust provides an infrastructure that current federated identity systems can rely upon for wide scale trust management.

### D. Reputation

Today, consumers visiting an e-commerce shopping site, citizens seeking services from their government, and people seeking social relationships through the Internet have little to guide them in assessing the trustworthiness of the parties they encounter. Reputation systems, as typified by the existing systems in use by eBay, ePinions, Slashdot, and others, appear to have considerable potential to fill this need. A recent paper by Masum and Zhang [14] provides a good survey of existing reputation systems and argues that pervasive reputation services are an essential building block for a mature digital society.

<sup>1</sup>Although federated identity systems use the term “service provider” we use “application provider” to avoid confusion with the trust and authorization service providers discussed earlier.

The challenge is in taking proven ideas from existing systems, possibly in combination with new ones, and extending them to a global scale. A global reputation network will necessarily be much more open and loosely connected, and it will need to provide for a wide variety of trust contexts.

$x$ Trust appears well suited to building such a network. Context selection in the  $x$ Trust system allows trust links to be established for the purpose of gathering informed opinions in various situations. Because individual  $x$ Trust Domains can solicit opinions and consolidate results as they see fit,  $x$ Trust provides a vehicle for deploying a wide range of trust formalisms, collaborative filtering techniques, recommender systems, and so on, each tailored to the needs of the organization operating the  $x$ Trust Domain. Hop-by-hop semantic mapping would allow these disparate techniques to interact in a loosely coupled, highly dynamic environment. The degree of interoperability that could be achieved in this way is not yet clear and will be one focus of future research.

### E. Secure messaging

There are two ways that  $x$ Trust can be used to implement secure messaging. One approach utilizes an existing infrastructure for secure messaging, such as S/MIME e-mail, relying on  $x$ Trust for certificate validation as we described in Section III-A. Alternatively, because  $x$ Trust places no restrictions on the content of a request, it is just as easy to ask a trusted  $x$ Trust Domain to deliver information to a remote destination as to request information from it. As with any  $x$ Trust request, the responder must recognize and know how to process it. In this case, the responder provides a mailbox into which the sender can place a message.

In both approaches, a path through the  $x$ Trust network will be associated with the authentication of the sender. In the first approach, the authentication path is the path through which the recipient retrieved the sender's public key to validate the signature on the message. In the latter approach, it is the path through which the message itself is routed. Trust relationships on this path are bi-directional (represented formally as reciprocal unidirectional trust links), where the sender trusts the receiver to properly authenticate and forward the message, and the receiver trusts the sender to not misrepresent the origin of the message.

Either way, as messages are delivered, they can be labeled with a value indicating the overall trustworthiness of the  $x$ Trust path. Messages originating from partners directly connected by an  $x$ Trust link would be labeled as highly trustworthy. Where direct links do not exist, messages (or keys) may still be routed through intermediaries. Messages with paths that pass through a short chain of partners might be labeled as moderately trustworthy, while those that pass through public TSPs would be labeled as relatively untrustworthy. A messaging client could easily sort or filter incoming messages by trust level, automatically giving priority to messages from more highly trusted (which will often imply highly valued) partners. Such a network has the nice property of providing automatic referrals. Alice may not be known to Carl, but if she has a close

relationship with Bob, who in turn has a close relationship with Carl, then she can send a message that is authenticated through Bob and will be labeled as highly trustworthy. It is not necessary for Alice to even be aware of these opportunities, as it is Carl's use of  $x$ Trust that provides the referral.

Domains participating in a trust network to relay messages (the second approach) can be configured in a variety of ways. For example, trust links for the messaging context may be specialized according to the level of authentication they demand of the sender. A link may be defined only for messages from public key authenticated senders, or from password authenticated senders. All messages delivered through a network defined by such links will be fully authenticated, even those from senders who are previously unknown to the recipient. Alternatively, a Domain may choose to define trust links for messages from anonymous senders, which would be routed only through Domains having trust links defined for this context. An anonymous message would reach its destination only if there existed a path through intermediate Domains all of which defined trust links for the anonymous message context. In this way, specialized message networks may be defined for different purposes and different assurance levels. All of these networks may be active simultaneously, with individual Domains defining multiple links and multiple contexts for each link.  $x$ Trust routing and context selection will ensure that the message is delivered by the most trustworthy route appropriate for the context.

This allows for many variations. Consider, for example, a situation in which a citizen wishes to send a message anonymously to a politician. For concreteness (and because the authors are Canadian), let us assume that Bob would like to send a message to Alice, the *Member of Parliament* (MP) for his *riding*. Suppose Bob selects the special `mail-to-MP` context for the message (in Canada, this context is already given special status: no postage is required for letters to MPs). Bob already has a trust relationship with the federal government for such purposes as paying taxes. Part of creating this trust link could involve the establishment of the `mail-to-MP` context. As part of the semantics of the trust relationship, Bob can request that his identity be stripped. The  $x$ Trust responder in this context authenticates Bob, confirms that he is currently registered in Alice's riding (the responder has access to the current voter list), then strips his identity and forwards the message to Alice. When Alice receives the message, she sees that it is from a citizen in her riding but does not know which one. She trusts the accuracy of the assertion because she received the message through a trust link configured specifically for this purpose.

Suppose, however, that for whatever reason Bob does not trust his government enough to be confident that his identity will be protected. In that case, he would not use the `mail-to-MP` context for his trust link with the government. He may instead belong to a Riding Association or some other citizen's group that offers a similar service. As part of the provision of the service, the group would have negotiated a trust link with the government, which would have to trust

the group to perform the citizen-to-riding mapping correctly. Whether this is realistic in practice does not detract from the possible variations the example is intended to illustrate.

Clearly it is unreasonable to expect individuals, or even organizations, to negotiate and configure a large number of trust links for such specialized purposes. In general, a small number of links would be active, some to general purpose TSPs and a few more specialized ones (*e.g.*, to employer, government, church, family, and close friends). The example of anonymous mail to an MP is meant to illustrate the potential to make specialized arrangements for unique services that utilize the existing trust network, without the need to negotiate globally uniform agreements or semantics.

#### IV. RELATED WORK

As we discussed in the introduction,  $x$ Trust incorporates ideas from many different areas. Some relate to core security technologies such as authentication and authorization. Others, such as techniques for semantic mapping, are less common in the security community. Because  $x$ Trust leaves so much to the discretion of individual Domains, it also provides a platform for combining a diverse range of supporting technologies. This section reviews work that relates directly to the core  $x$ Trust ideas, and also highlights some complementary technologies that could be used by individual Domains to implement their services.

##### A. X.509

The X.509/PKIX [9] public key infrastructure is perhaps the most successful large-scale security infrastructure currently deployed, yet it has been unable to reach the pervasive scale it once promised.  $x$ Trust addresses several issues that we believe have prevented more applications from using the X.509 PKI.

Although the X.509 PKI relies heavily on some kind of directory service to publish certificates, the emergence of a global publishing system (*i.e.* the World Wide Web) has not enabled widespread PKI deployment. By publishing digitally signed objects through an untrusted medium, the X.509 PKI intrinsically creates a delay between when a subject's information is "blessed" by an authority and when a relying party uses that information. This delay is controlled by the authority, who decides when to publish updates, yet different relying parties will have their own levels of sensitivity to any delay. The authority must choose its publishing schedule, which is unlikely to meet the needs of all, or even most, relying parties. In  $x$ Trust, each relying party adheres to its own freshness requirements by deciding whether to use cached data or to submit a new query to the  $x$ Trust network.

Using an untrusted publishing medium allows the X.509 PKI to keep authority credentials offline, whereas  $x$ Trust uses active authorities that participate in interactive protocols with other entities. We find X.509's offline property to be a dubious advantage at best. Consider, for example, that the keys of TLS-enabled servers (web or otherwise) are rarely compromised, even when the servers themselves are compromised. Going to great lengths to keep an authority's keys offline does not seem

to provide much return for the effort, and the advantages of  $x$ Trust's online authorities are compelling.

The X.509 PKI has some degree of context sensitivity in that a certificate can contain "policy" identifiers specifying its purposes. In theory, a single user can have several certificates, each issued under a different policy for use in a different application. This could approximate some degree of  $x$ Trust's context awareness, however in practice multiple application-specific certificates are rare.  $x$ Trust also provides more contextual flexibility, because  $x$ Trust contexts can be arbitrarily defined according to any combination of parameters – for example, not just the application in question, but also who is performing the application, and when.  $x$ Trust contexts can also be dynamically redefined at any time. This flexibility could only be matched by an X.509 authority issuing and revoking certificates very dynamically (essentially placing the CA's keys online).

Even if dynamically-issued, context-sensitive X.509 certificates were the norm,  $x$ Trust still provides an advantage for inter-domain trust management in that  $x$ Trust distributes the burden of trust path discovery and validation. The X.509 PKI requires that clients be able to access the entire repository of all published certificates, and that they sift through the repository to discover the paths they need.  $x$ Trust does not require any such global access or processing, and so individual elements in  $x$ Trust are much simpler to implement and manage. Furthermore, the X.509 PKI has rigid, limited rules for mapping semantics between domains, as these rules must be implemented by every X.509 PKI application.  $x$ Trust abstracts and distributes mappings away from clients, simplifying them even further.

One last note about PKIs. Although recent initiatives for delegated path validation (DPV) in PKI aim to relieve client PKI applications of PKI's complexities [17], [7], there is as yet no defined *infrastructure* for DPV servers to work with. Current DPV solutions seem a stopgap measure at best. However, DPV servers can use  $x$ Trust for certificate validation to create a scalable, manageable X.509 PKI.

##### B. Social Networks and Trust Networks

Social networks have received considerable attention in recent years. Many services, as represented by Friendster and FriendFinder, exist purely for the purpose of building a network of contacts. In general, these networks are not context sensitive and they are not leveraged for purposes other than social communication.

The LinkedIn service [10] is similar, but specialized for building a professional network and using it for business purposes. It strives for high quality connections by placing tight limits on the use of transitivity, relying on a process of invitation and referral for growth and cross linking of sub-networks. (We note in passing that when negotiating trust links,  $x$ Trust Domains are free to follow a similar process for their own trust management.) Connections in LinkedIn are not explicitly context sensitive, but sub-networks tend to form

around specific business contexts because their members tend to know each other for similar reasons.

Other social networks take a more sophisticated approach to helping users or members identify other like-minded individuals. For example, Acorn [13] is a multi-agent system in which personal agents represent the interests of a person and actively seek out other agents representing people with similar interests. Yenta [6], [5] performs a similar function, but attempts to simplify the process of describing a user's interests by automatically scanning their personal files to build a profile of interests. Yenta also goes to considerable lengths to provide a secure, trustworthy environment. As with  $x$ Trust, all communication is encrypted and mutually authenticated by default. Yenta relies on TLS for this purpose, using self-signed certificates for authentication. This eliminates dependence on third party authorities, but provides only weak protection against man-in-the-middle attacks.

These systems generally focus on discovery of relationships and less on the use of trusted paths, whereas  $x$ Trust relies on people to form their own relationships, then utilizes the resulting network to provide trust assurances for a wide range of applications. It would be interesting to combine several of these ideas. For example, one could combine techniques like those used in Yenta or Acorn with the referral process of LinkedIn to help users discover relationships that will enable them to incrementally add useful partnerships to their  $x$ Trust network.

A number of well-known systems implement trust networks that are similar to the  $x$ Trust approach. Pretty Good Privacy (PGP) [15] users can establish a "web of trust" by signing each other's keys. A user can select a set of other users to be "trusted introducers" – PGP users that are trusted to provide valid keys for other users – or "meta-introducers" (who are trusted to vouch for other introducers). PGP users can also assign a "level of trust" to trusted introducers. For all this sophistication, PGP lacks any notion of context in its web of trust. Although a user could keep different "keyrings" for different contexts, PGP does not distinguish between keys signed for different purposes. PGP also imposes a single trust model on all of its users, forcing all users to distill their trust into one of three levels – untrusted, marginally trusted, and fully trusted. In contrast,  $x$ Trust is fully context-sensitive and allows users to employ different trust models.

The Web of Trust concept used by the ePinions rating site creates a reputation network that is specialized for the purpose of providing product and service ratings and reviews. Members of ePinions explicitly manage their own connections to the trust web, and links in the web are unidirectional. Unlike  $x$ Trust, however, the ePinions Web of Trust does not provide context sensitivity. Trust is placed in the general competence and credibility of reviewers but not, for example, in their expertise in specific domains.

The Web-o-Trust project [22] has created an e-mail (SMTP) whitelisting network with a similar trust model. Individuals create their own whitelist of trusted SMTP senders which can include specific addresses and may also reference other

whitelists. It is possible to trust the complete transitive closure of the network, or to limit trust to a small region. Again, the network embodies the unidirectional property. However, unlike  $x$ Trust, absolute trust is not required: a traversal limit may be specified when referencing another list, limiting the scope of trust transitivity. This is possible because, although storage of whitelists is decentralized, processing of a whitelist *is* centralized for any given transaction. In  $x$ Trust, peers must be trusted to respect transitivity constraints because processing of a request is distributed along the request path.

### C. Onion Routing

Onion routing is a communications infrastructure that is intended to offer resistance to eavesdropping and traffic analysis [8]. It is similar to  $x$ Trust in that it defines a network through which communication is routed in a way that separates identification from routing. The goals of the two systems, however, are in a sense completely opposite. Connections in an onion routing network are always anonymous while communication need not be. Connections in an  $x$ Trust network are always fully authenticated, while communication need not be. It would be useful to combine the strengths of both systems to produce a system capable of providing highly trustworthy anonymization and traffic masking services. Such a system would, for example, be able to more directly address needs such as those described in the example of Section III-E of anonymous mail to a government official.

It is not obvious, however, how to marry the two systems. Onion routing depends strongly on the ability of the first node in the route to accurately identify the complete path through the network (a property commonly referred to as *source routing*). The design philosophy of  $x$ Trust, in contrast, is to make local decisions only, trusting peers to service requests in whatever way they deem most appropriate.  $x$ Trust's scaling properties are derived from this philosophy, so it is not desirable to alter it. This is an interesting area for further investigation.

### D. Other Areas

KeyNote [2], [1] is a method for "specifying and interpreting security policies, credentials, and relationships." Although it is called a "trust management" system, KeyNote focuses on the definition and execution of security policy within a single application. That is, it allows a single entity to "manage" its own security through a powerful syntax and semantics. However, KeyNote does not address the issues that arise when dealing with Principals from foreign Domains that may use different semantics and/or may not be directly trusted.  $x$ Trust complements and completes KeyNote, allowing it to function in these multi-Domain situations. An  $x$ Trust Domain can use KeyNote to express and enforce its security policies, and that Domain's KeyNote compliance-checker can rely on  $x$ Trust to obtain credentials and attribute values for input into its decision-making machinery. Put another way, KeyNote can be seen as a method for configuring an  $x$ Trust Domain. Together,

KeyNote and  $\mathcal{X}$ Trust create a powerful multi-domain trust management system.

The Spectrum project [21] is developing several techniques that can be used for evaluating trust links in different contexts. Spectrum’s trust inference engine could be used to help establish metrics for trust routing, and a Domain administrator could use Spectrum’s belief visualization methods to understand and, possibly, manually adjust trust in different contexts.

The Simple Distributed Security Infrastructure (SDSI, or “sudsy”) [18] and the Simple Public Key Infrastructure (SPKI, or “spooky”) [4] suffer from scalability problems that  $\mathcal{X}$ Trust avoids. In SDSI/SPKI, a client Principal that needs access to a resource must obtain an appropriate authorization certificate, one that the resource’s guardian will accept. Thus, a client is forced to contact the *resource’s* trusted peers to obtain the proper credentials. This means that any authority for the client must also be directly trusted by the resource, and gives rise to the problem of agreeing on a common set of semantics and trusted authorities. Furthermore, SDSI/SPKI’s “local names” are of limited use: a name like (alice’s bob’s carl) is only meaningful to alice and those who know her. Such a name also represents a SDSI/SPKI trust path, which further limits the name’s utility. These limitations make it difficult to use SDSI/SPKI in heterogeneous, multi-domain networks.

## V. CONCLUSIONS

We have described  $\mathcal{X}$ Trust, a general purpose security framework that supports a wide range of applications and security mechanisms, and does so in a way that makes trust visible. Interaction and communication between peers in the network is mediated through trust links that explicitly identify what each party in a transaction trusts the other to do.  $\mathcal{X}$ Trust provides basic services for identifying paths between peers that are trusted for specific purposes, and for utilizing those paths for the trustworthy exchange of information. All exchanges are encrypted and authenticated by default.

The design philosophy of the system has four general principles. First, trust is always context sensitive. Every transaction in  $\mathcal{X}$ Trust specifies a context, and communication is only possible when a path exists between peers in that context. Second,  $\mathcal{X}$ Trust abstracts trust services from the applications that use them. Third, trust relationships are always expressed using semantics local to a specific Domain in the network. Domains are free to share semantics as broadly as they wish, but there is no requirement to do so. Together, the second and third design principles make  $\mathcal{X}$ Trust highly adaptable to different trust environments. Finally, trust is absolute. One’s trusted peers are fully responsible for ensuring that trust is not betrayed.  $\mathcal{X}$ Trust is therefore designed so that relying parties only depend on entities with which they have a formal relationship.

## ACKNOWLEDGMENTS

The authors would like to thank Andrew Nash, who first suggested that routing protocols might be helpful in PKI certificate path discovery, and John Linn, for much helpful

discussion, particularly with respect to federated identity systems.

## REFERENCES

- [1] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The KeyNote Trust-Management System Version 2. The Internet Engineering Task Force, RFC 2704, September 1999. See also <http://www.cis.upenn.edu/~angelos/keynote.html>.
- [2] Matt Blaze, Joan Feigenbaum, and Angelos D. Keromytis. KeyNote: Trust Management for Public-Key Infrastructures. In *Proceedings of the 1998 Security Protocols International Workshop*, pages 55–63, Cambridge, England, April 1998. Springer LNCS vol. 1550. .
- [3] Paolo Crucitti, Vito Latora, Massimo Marchiori, and Andrea Rapisarda. Efficiency of Scale-Free Networks: Error and Attack Tolerance. In *Physica A* 320, pages 622–642. Elsevier Science B.V., 2002. Retrieved August 19, 2004 from [http://www.w3.org/People/Massimo/papers/2003/tolerance\\_physicaa\\_03.pdf](http://www.w3.org/People/Massimo/papers/2003/tolerance_physicaa_03.pdf).
- [4] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI Certificate Theory. The Internet Engineering Task Force, RFC 2693, September 1999. See also <http://world.std.com/~cme/html/spki.html>.
- [5] Leonard N. Foner. Yenta: A Multi-Agent, Referral-Based Matchmaking System. In *Proceedings of the First International Conference on Autonomous Agents (Agents ’97)*, Marina del Rey, CA, February 1997.
- [6] Leonard Newton Foner. *Political Artifacts and Personal Privacy: The Yenta Multi-Agent Distributed Matchmaking System*. PhD thesis, MIT Media Lab, Massachusetts Institute of Technology, June 1999.
- [7] Warwick Ford, Phillip Hallam-Baker, Barbara Fox, Blair Dillaway, Brian LaMacchia, Jeremy Epstein, and Joe Lapp. XML Key Management Specification (XKMS). Technical report, The World Wide Web Consortium, March 30 2001. Retrieved August 19, 2004 from <http://www.w3.org/TR/xkms/>.
- [8] David Goldschlag, Michael Reed, and Paul Syverson. Onion Routing for anonymous and private internet connections. *Communications of the ACM*, 42(2):39–41, February 1999.
- [9] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. The Internet Engineering Task Force, RFC 3280, April 2002. See also <http://www.ietf.org/html.charters/pkix-charter.html>.
- [10] LinkedIn. <http://www.linkedin.com/>.
- [11] John Linn, editor. *Liberty Trust Models Guidelines*. Liberty Alliance Project, July 25 2003. Retrieved August 19, 2004 from <http://www.projectliberty.org/specs/draft-lib-tsp-trust-models-v1.0-15.pdf>.
- [12] Eve Maler, Prateek Mishra, and Rob Philpott, editors. *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1*. Organization for the Advancement of Structured Information Standards (OASIS), September 2 2003. Retrieved August 19, 2004 from <http://www.oasis-open.org/committees/download.php/3406/oasis-20ssts-saml-core-1.1.pdf>.
- [13] Stephen Marsh, Ali. A. Ghorbani, and Virendra C. Bhavsar. The ACORN Multi-Agent System. *Web Intelligence and Agent Systems*, 1(1):1–21, March 2003.
- [14] Hassan Masum and Yi-Cheng Zhang. Manifesto for the Reputation Society. *First Monday*, 9(7), July 5 2004. Retrieved August 18, 2004 from [http://www.firstmonday.dk/issues/issue9\\_7/masum/index.html](http://www.firstmonday.dk/issues/issue9_7/masum/index.html).
- [15] Network Associates, Inc. *PGP 7.0 User’s Guide*, January 2001. Retrieved 19 August 2004 from <http://www.pgpi.org/doc/guide/7.0/en/>.
- [16] Radia Perlman. *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols*, chapter 12, pages 337–338. Addison-Wesley, 2nd edition, 2000.
- [17] D. Pinkas and R. Housley. Delegated Path Validation and Delegated Path Discovery Protocol Requirements. The Internet Engineering Task Force, RFC 3379, September 2002.
- [18] Ronald Rivest. SDSI – A Simple Distributed Security Infrastructure. Keynote address at the Sixth USENIX Security Symposium. See also <http://theory.lcs.mit.edu/~cis/sdsi.html>, July 24 1996.
- [19] Audun Jøsang and S. Lo Presti. Analysing the Relationship Between Risk and Trust. In T. Dimitrakos, editor, *Proceedings of the Second International Conference on Trust Management*, April 2004.
- [20] The Semantic Web. <http://www.w3.org/2001/sw/>.
- [21] The Spectrum Project. <http://security.dstc.com/spectrum/>.
- [22] The Web-o-Trust Project. <http://www.web-o-trust.org/>.