Poblano
A Distributed Trust Model for Peer-to-Peer Networks

Rita Chen and William Yeager,
Sun Microsystems

1. Introduction

Trust is at the core of most relationships between human beings. We all have a sense of what it means to trust someone. The parameters of trust are often personal, and thus, decentralization is the nature of trust, because each individual has his/her own opinions.

On a decentralized network, like Peer-to-Peer (P2P), users can see from where the information arrives, as well communicate their opinions on both the information they have acquired, and the peers who are its source. These personal opinions can be collected, exchanged, and evaluated. Furthermore, these opinions, when evaluated, can be used as guidelines for searching for information, and recommending information sources, thus, creating decentralized, personalized "Webs of Trust."

When such a decentralized trust model is implemented on a P2P topology, trust between peers begins to mirror those real-world relationships with which users are familiar, and permits software engineers to craft interfaces to the underlying trust model that are both understandable and usable. Trust becomes a social contract with social implications for the participants. Each such peer will develop a reputation among his peers, which is the basis for P2P trust relationships.

The goal of this project is to establish such a decentralized trust model on the Project JXTA platform. The model presents not only trust relationships between peers but also relationships between peers and codat (Section 2). The protocols used to disseminate trust are proposed, and the algorithms used to update trust are described (Section 3). One of the initial applications of this model is to perform reputation guided searching.

A second application of Poblano is to build a recommendation system for security purposes. This is described in the second part of this paper (Sections 4-7). In the longer term, these distributed relationships can be widely adopted by applications for which trust can be based on the norm for social interaction between the participating peers.

2. Representing Trust Relationships

What we describe below is the first steps towards implementing a model to help us understand distributed, P2P trust. For an implementation to be achievable, we must begin with programmable concepts. Poblano is much like your first experience with spicy food. It is just the first mild taste of the hot decentralized network.

In current trust or reputation models, such as those used in Free Haven and Publius projects [1][2], the degree of trust is calculated with parameters such as performance, honesty, reliability, etc..., of a given peer. If a peer cheats at playing cards, for example, the peer might be deprived of his ability to authenticate and join another card game.

But for the group of people interested in cooking, the above measurement is too biased towards personal risk, and not content or codat, and will be of little use. Hence, for the cooking group, the trust should be biased towards data relevance, or the quality of recipes. For us trust has multiple components or factors, and we are looking at a factor of trust which is based on the group's interests, or group content relevance.

In order to collect the information on a group's "interests," Poblano introduces the codat trust component, as shown in the following figure:

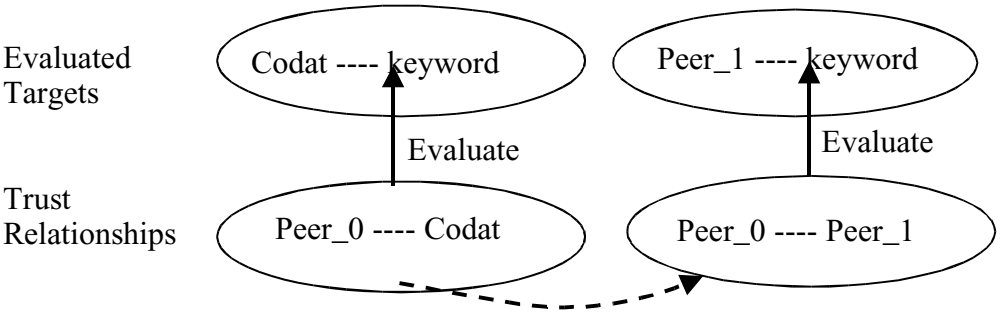


Fig. 1. Trust Relationships

At first, a user (Peer_0) needs to evaluate his trust on a codat to build a trust relationship with that codat. In order to evaluate trust with respect to the peer's interests, the latter are represented as keywords. Then, the results of the "interests" evalution on the set of codat that Peer_0 received from Peer_1 will be used to evaluate the Peer_0's trust of Peer_1 as a source for the given collection of keywords.

In this way, the evaluation of the trust for the peer is based on the user's interests or keywords. Again, the codat trust component is different than the traditional trust concept, which is identified as the risk trust component in Poblano. The risk trust component's value will be at least determined by codat integrity, e. g., the codat contained a virus as noted by a virus pre-processor, peer accessibility (is the peer up

most of the time), and peer performance (long delays in retrieving data).

To represent the different trust components Poblano uses following classes:

1. CodatConfidence - This class is to evaluate the codat trust component according to a keyword. In the class "CodatConfidence", there are 4 elements: keyword, codatID, local flag for marking codat either local or remote, and confidence value. Initially, the confidence value has two metrics: popularity, and relevance to keywords. Popularity is monotonically increasing and is incremented at the provider each time the codat is requested. The relevance is in the range (-1, 0, 1, 2, 3, 4) as described below[1].

2. PeerConfidence - This class is to evaluate the codat peer trust component according to a keyword. In the class "PeerConfidence", there are 3 elements: keyword, peerID and confidence value. In addition to the CodatConfidence metrics above, we additionally keep the running average of the popularity of each codat accessed from this peer[2] for the given keyword.

3. Risk - This class is to evaluate the peer's risk trust component. Its elements are currently peerID, integrity of the codat, accessibility and performance.

Among these three classes, PeerConfidence and Risk are the classes of most interest. They will be used to determine if the target peer is able to cooperate (the cooperation threshold is described in a later section) and is, thus, trustworthy.

CodatConfidence can be seen as the means by which we calulate PeerConfidence from search results (keyword relevance), and user evaluation. For a user, rating codat is easier than rating a peer in a general sense.

Hence, Poblano will provide a API to develop GUIs for codat user_rating, i. e., implicitly a user has a sense of how well the retrieved codat fits the search criteria. This goes beyond keyword match and is most easily provided by user input, and will be the user supplied factor of the CodatConfidence relevance metric.

Note, there is a reason why the word "Confidence" is used here instead of "Trust." PeerConfidence is the codat peer trust component of user interest. Trust is thus the "sum" of PeerConfidence and Risk. To keep the components clearly separated we appended "Confidence" to the names of the first two classes.

The next section will focus on how to distribute the classes, how to update their values and make them meaningful to different applications.

[1] See section 3 for how these values are calculated.
[2] Ibid.

3. Distributing Trust Relationships

3.1 Processing Trust Relationships

Each peer will have PeerConfidence tables and CodatConfidence tables. Since codat is associated with peer groups, there is one CodatConfidence table for each group. If the peer belongs to more than one peergroup, it will have more than one CodatConfidence table. The first PeerConfidence table is for those peers in a given peergroup for which the peer has keyword, codat information. The second PeerConfidence table is across all the peer groups to which the peer belongs. The latter table permits us to calculate a peergroup independent PeerConfidence value.

Given a Peer, then for each PeerGroup(i), there will be a CodatConf Table recording the keyword, codat relationships:

PeerGroup(i): CodatConf Tables

Keyword_0	Keyword_1	...	Keyword_n
CodatConf_0,0	CodatConf_1,0	...	CodatConf_n,0
CodatConf_0,1	CodatConf_1,1	...	CodatConf_n,1
...	...		...
CodatConf_0,j	CodatConf_1,k	...	CodatConf_n,m

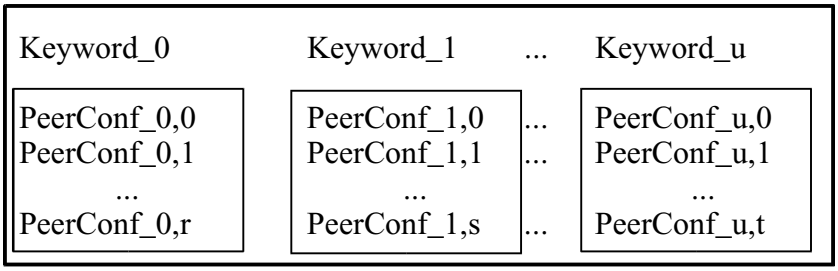
And, for each such PeerGroup(i), there will be a peer confidence table derived from the above table:

PeerGroup(i): PeerConf Table

Keyword_0	Keyword_1	...	Keyword_n
PeerConf_0,0	PeerConf_1,0	...	PeerConf_n,0
PeerConf_0,1	PeerConf_1,1	...	PeerConf_n,1
...	...		...
PeerConf_0,r	PeerConf_1,s	...	PeerConf_n,t

Finally, we have the table for PeerGroup independent peerConfidence:

PeerGroup Independent PeerConf Table



When a peer does a search on a keyword, it will do the following:

Step 1) Lookup the keyword in its own CodatConfidence table. If there is a local codat associated with the keyword, i. e., the local flag is true, then succeed[3]; else

Step 2) Lookup the keyword in its PeerConfidence table. If there are Peers highly associated with the keyword, forward the request to those peers. Each of those peers will perform step 1 and 2 using their own tables again. If such a peer has a valuable codat related with the keyword, the peer will inform the requesting peer, and terminate the search.

If the lookup fails, then the peer can resort to the peergroups' keyword tables discussed below for other possible sources.

Step 3) Calculating the CodatConfidence: The requester accesses the codat. Through clicking on the codat, the provider's popularity value for the codat will be increased by 1. By the means of the requester's rating of the codat, the old confidence object for the provider with respect to the keyword will be updated. Now the requester has an entry added in his own CodatConfidence table. The collection of such codat ratings for the given peer will be used to generate a PeerConfidence value for the provider with respect to the given keyword, codat pair. This is from the requester's eyes. The same codat rating will be also fed back to the provider. The provider may update his CodatConfidence table based as a function of the previous value, feedback on codat, and his confidence in the requester. This is from the provider's eyes.

More details will be described in the next sub-sections. The above is an example to show why a peer needs these two kinds of confidence tables and how they're related.

In general peers will not be members of all PeerGroups, and new peers will not belong to any PeerGroup. Since PeerGroup membership is motivated by keyword interest, for those PeerGroups which permit it, such a peers should be able to retrieve

[3] There will be instances when a peer will wish to search remotely even if the local codat is good.

PeerConfidence tables from these groups.

3.2 Propagating Trust Relationships

The above sub-section gives us an overview of how the two components of trust relationships, Confidence and Risk, map to hard coded information. This sub-section will discuss how to propagate such information to make a complex chain of relationships.

The following is a simple algorithm to rate a propagated "degree of trust." Here, a peer can have a trust value of -1, 0, 1, 2, 3, or 4 [3]:

Value	*Meaning*
-1	Distrust
0	Ignore
1	Minimal trust
2	Average trust
3	Good trust
4	Complete trust

Table 1. Trust Value

For the trust values of 0 and -1, the associated codat is never accessed. The trust value will be propagated through a transaction pipe. The trust value of a target for a single path, $V_{path}(T)$, from peer S to peer T through peers P_i, i= 1, 2, ..., n, can be calculated as follows:

Where $V(P_i)$ is greater than or equal to 1,

$$V_{path}(T) = \frac{1}{4n}\left(\sum_{i=1}^{n} V(P_i)\right) \times V(T)$$

Formula 1

Here, $V(P_i)$ is the trust value of the peer, P_i who provides the information. $V(T)$ is the trust value on the target peer, T.

For multiple paths, the final trust value is the average of all the propagated trust values.

Here is an example to illustrate the model. Assume: there are two paths from peer A to peer D. The first path is through peer B and C, the second one is though B, E and F. C trusts D with a value of 3, B trusts C as a recommender with a value 2, and A trusts B as recommender with a value of 3. So,

$$V_1(D) = \frac{V(B)+V(C)}{8} \times V(D) = \frac{3+2}{8} \times 3 = 1.88$$

By using the same algorithm, assume the trust value of the second path $V_2(D)$ is 2.15. Then, the trust value A gives D is the average of two paths, 2.01. Besides propagation, reputation can be initialized and updated.

In Poblano, we have the two values for PeerConfidence and CodatConfidence. The CodatConfidence value is the information to be propagated and the PeerConfidence value is the carrier information to be used for weighting. So, the above equation can be transformed as follows where the CodatConf and PeerConf are the *relevance* metrics for codat within a given PeerGroup:

$$CodatConf_{path} = \frac{1}{4n}\left(\sum_{i=1}^{n} PeerConf(P_i)\right) \times CodatConf$$

Formula 2

Overall, there are two situations when we will need such propagation. The first is when requesting information remotely and successfully. The provider needs to send the CodatConfidence *object* to the requester. If after computing the above $CodatConf_{path}$, the requester wants the codat, then the codat itself must be sent to the requester. Even if the codat transfer occurs between P_1 and P_n, the $CodataConf_{path}$ remains as if the data was received through the pipe.

The second situation is when giving feedback to codat providers. The updated CodatConfidence object from requester will be propagated back to the provider.

3.3 Updating Trust Relationships

Trust value updating is much more complex than propagation. This is because each peer needs to update three kinds of confidence tables. Also, the updates are based on a peer's rating as well as on the feedback rating. Let's describe these updating strategies one at a time beginning with the simplest:

Case 1) A peer wants to update his old CodatConfidence using his own rating and the CodatConfidence propagated from remote peers. The propagated popularity will be a running average. We will focus on the confidence and quality updating for now. The updating function is:

$$\begin{aligned} new\_CodatConf &= F(old\_CodatConf, propagated\_CodatConf, user\_rating) \\ &= a \times old\_CodatConf + b \times propagated\_CodatConf + c \times user\_rating, \end{aligned}$$

Formula 3

$a+b+c = 1$; a, b and c non-negative real numbers. Currently, $c = .70$ since the user's personal rating is the most important criteria for a user.

If the new popularity value > old popularity value, then $a = .10$ and $b = .20$, otherwise, $a = .20$ and $b = .10$. More popular codat receives a slight edge.

Here, as mentioned in section 3.1, the user_rating is from user input. In some cases, neither the old_CodatConfidence nor the propagated_CodatConfidence is available. In this case they must be preset to 1. The same rule can be applied to the following functions.

Case 2) A peer wants to update his old CodatConfidence using a feedback. He may also have the PeerConfidence on the peer who gives the feedback ("feedbacker"). A feedback actually is a reverse-propagated CodatConfidence from a feedbacker. The updating function is:

$$\begin{aligned} new\_CodatConf &= F(old\_CodatConf, feedback, PeerConfoffeedbacker) \\ &= (old\_CodatConf + feedback \times \frac{PeerConf_{feedbacker}}{4})/2 \end{aligned}$$

Formula 4

In most of cases, the peer doesn't have PeerConfidence for the feedbacker, so this parameter can be preset to 1.

Case 3) A peer wants to update the PeerConfidence of an information provider in a peerGroup. Here, nobody tells this peer how the provider performs. The peer has to generate his own opinion on the provider, associated with one or more keywords. What he knows is the Codat Confidence, relevance metric of all the codat the provider provides. So the updating function is:

$$\text{new\_PeerConf} = F(\text{old\_PeerConf}, \text{set of CodatConf related to the provider})$$

$$= (\text{old\_PeerConf} + \frac{1}{|K|}\sum_{a \in K} \text{CodatConf}_{\text{provider}})/2$$

$|K|$ = number of keywords, a in K, related to the provider.

Formula 5

Given that there are so many different updating functions, a basic Bayesian belief network approach may be adopted for them later. Via Baye's formula we can use current data to derive what the posterior model looks like. This could be closer to reality.

3.4 Calculating Cooperation Threshold

To make these trust values meaningful for users, Poblano introduces the cooperation threshold. If the PeerConfidence value is greater than the threshold, the peer is considered to be cooperative. Otherwise, the cooperation is too risky to the user. The calculation of the threshold is based on the risk value, CodatConfidence values and importance value. The importance value is a new concept here. It measures how important the cooperation is to the user. Sometime a user may be willing to take a risk, i. e., override the Trust model's recommendation, even the PeerConfidence may be low. Importance has a value of (-1, 0, 1, 2, 3, 4) and should be input by users through a GUI. The default value is 2.

The risk value is in the range from 0 to 4. 0 means no risk and 4 maximum risk. It is a statistically computed result on the peer accessibility and performance. The traditional network quality of service study method will be adopted here [1][2]. The default value for risk is 2 or medium risk. Therefore, if the following rule is established, then cooperation is possible:

$$PeerConf_{keyword} \times Importance > \frac{Risk_{peer}}{\frac{1}{|K|}\sum_{a \in K} CodatConf_{peer}}$$

Formula 6

Here, K is the collection of all keywords, a, for the given peer for which we have CodatConfidence values across all peergroups, and |K| is the number of such keywords.

The CodatConfidence values on the peer (related to a keyword K) are used to represent the experienced competence. Assuming the importance is a constant, if the risk is high and the experience is bad, the threshold will be high. In this case, the PeerConfidence may not be higher than the threshold.

4. Security and the Poblano Trust Model

The security "mantra" is privacy, authentication, integrity, and non-repudiation. There are cryptographic algorithms and protocols that one can implement to guarantee that a conversation is private, authenticate a user, insure the integrity of data, and to assure that a transaction cannot be repudiated by its originator. These algorithms are well covered in the literature, see for example [5], and will not be discussed here. To the above cryptographic list, one can add secure access to codat[4], or authorization. The model we propoose below will not present a specific authorization solution but rather be open enough to permit the implementation of secure, codat access schemes. Our challenge is to create a model for P2P, distributed security in which all of the above can be deployed.

Essential, and at the very core of our model is the ability to create, and distribute public keys given a peer-generated, private-public key pair. In the tradtional Internet security paradigm, certificate creation is done using certificate authorities (CA) whose signature appended to a certificate guarantees the certificate's content for any recepient that has secure access to the CA's public key. In most cases this latter public key resides in a root certificate on the recepient's system.

Almost all of what we discuss in the following sections is about how to create and distribute signed certificates in a P2P network. The goal is not to create a true, distributed Public Key Infrastructure (PKI), but rather, by using the Poblano Trust Model, how to create a *trust spectrum* which neither requires nor prohibits the presence of a PKI.

Given such a trust spectrum, this is followed by a discussion on establishing unique peer identities to permit authentication, and the assignment of the peer's associated access policies within a peerGroup, i. e., authentication and authorization. We finish by considering the topic of mobile credentials, or how to make a system's private security creditials securely available. This latter discussion is an introduction to what is being done in IETF Security Area's Secure Availability of Credentials Working Group on this topic [6, 7].

[4] Codat is a general term that covers static as well as dynamic or executable data, which is locally or remotely stored. One can add to this defintion abstractions like routes codat might take, some of which might be priviledged.

4.1 Certificate Distribution and the Trust Model

We are all familiar with SSL and its IETF successor, TLS [8]. The implementations in use today require certificates to be signed by recognized, trusted certificate authorities to both establish identity, and exchange public keys for public-key algorithms like RSA, and Diffy-Hellman. In a p2p network we *cannot require* every participating peer to acquire, i. e., pay for, a CA signed certificate in order to implement, for example, p2p TLS. In fact, we want p2p zero-dollar-cost certificates.

P2p zero-dollar-cost certs will initially mean exchanging self-signed certificates, or certificates signed or cosigned by a trusted 3rd parties much like it is done with Pretty Good Privacy (PGP) [9] "Webs of Trust." At the same time we do not want to prohibit the very strong security that is currently used in the Internet.

So, we are proposing a *trust spectrum* that has CA signed certificates near one endpoint, and *self-signed certificates* near the other. At what spectrum point of trust a peergroup chooses to communicate is up to the participants in that group. We have no power over what function a peer performs in the network. Any peer can join a peergroup and offer its services. This includes recognized CA's. The peergroup members will assign a *level of trust or peerConfidence* to that peer, and to one another. And, that is where Polano comes into the picture. The details are discussed in section 4.2.

One might question the value of self-signed certificates. After all, users of these certificates are left open to the imposter-in-the-middle attack. If Bob receives Alice's self-signed certifcate in her security advertisement, Bob has no way to guarantee that in fact he received the certificate from Alice, and conversely, for Alice. The intruder, Jeanne, can be in the middle of a conversation seeing everything in clear text, and having given her "faked" self-signed certificates to both Bob and Alice, pretending to be both of them. Since Jeanne possesses both Bob's and Alice's public keys, her presence is undetectable. While it does take a great deal of effort to steal Alice's and Bob's peer identities, it can be done using advertised, public information and information acquired as the imposter-in-the-middle[5]. Such an intruder can fully participate in a peergroup using this stolen role.

But, we argue that for a certain class of applications, this behavior will be perfectly acceptable as long as the above threats are clearly understood by the users. For example, a family can form a peergroup, and do secure instant messaging among themselves. The underlying messages will be private, secured with TLS using 1024 bit RSA, 128 bit RC4, and SHA-1. The family may not worry that an imposter might try and watch their conversations. This is a cost/risk decision whose risk is extremely small.

[5] A great deal of work is required to steal a peer's identity, i. e., all of Alice's advertisements must be duplicated, knowledge of possible encrypted passwords and, and pipe end-point resolution spoofed. This is can be done with self-signed certifcates and this attack.

If the imposter-in-the-middle attack is an unacceptable risk, and p2p zero-dollar-cost certificates are desired, we then can move to a more secure spectrum point by exchanging certificates in person using infrared, or floppy disks, for example. This is eyeball-to-eyeball trust, and certainly, in a family sized peergroup, this is achievable and very secure.

We can increase security with cosigned certificates which are more difficult to forge since if Jennifer requires that all certificates she uses be cosigned by Luc, has Luc's public key on her system, and initiates a private communication with Zhiqun, Dan is the imposter in the middle, and forges Zhiqun's certificate that is cosigned by Luc, Dan will also will have had to forge Luc's certificate that is resident on Jennifer's system.

Still, if this is not secure enough, then peergroup members can delegate certificate signature authority to selected members of a peergroup. This is similar to applying the PGP "Web-of-Trust" to p2p. Traditionally, if Chloe wants to acquire a signed certificate from Mark who is a CA, she generates her public, private key pair, sends the public key, algorithm parameters and personal identification to Mark, and then proves ownership of the private key. The latter can be accomplished by Mark with a challenge encrypted in the public key and sent to Chloe who owns the private key. Only she can decrypt the challenge, again encrypt it in her private key, and send it back to Mark for verification. Once ownership is verified, Mark can issue a signed certificate to Chloe. To verify that Mark indeed signed the certificate Chloe must have Mark's public key. And, if Chloe wants to communicate securely with Steven, the he too must have Mark's public key and must trust Mark's signature. This makes the imposter-in-the-middle attack very difficult since Mark's signature is created with his private key, and Chloe, and Steven have Mark's public key. This taken with strong authentication, and authroization can prevent role theft.

The above can then be applied to create a "Web-of-Trust"-like signed certificate distribution in a peergroup. We can create a keyring of signed certificates and assign trust using personal input using the Poblano algorithms.

Still, if "personal trust" is not sufficient, we can strengthen our security by including root certificates with each release of the JXTA Project platform. This will permit a peergroup-CA, Bob, to both sign and distribute signed certificates to peergroup members. If Bob has signature authority in his peergroup, and Chloe is a peergroup member that trusts Bob, then she must have his public key on her system. The easiest way to accomplish this is to have a root certificate on each peer in the peergroup when the Project JXTA software is loaded. These root certificates can be generated on www.jxta.org or other trusted Project JXTA satellite organizations which may in fact be true certificate authorities.

Given such a zero-cost bootstrap mechanism, Bob can request a signed certificate from any of the trusted Project JXTA satellites, their goal being to propagate signing authority within peergroups without taking on the entire responsibility. Chloe can

then, in the same way, open a secure, TLS session with Bob's system receiving Bob's satellite-signed X.509v3 certificate in the TLS handshake, verifying Bob's autenticity, and procede to acquire her certificate signed by Bob using a totally secured connection. At the same time she save's Bob's certificate on her keyring for future use of his public key. Here, for example, she might want to send Bob some private email using S/MIME [10].

If required, Bob, and other peergroup-CA's can maintain certificate revocation lists to assure that any transaction with a known, breached certificate cannot take place, thus taking us one step closer to a true peergroup PKI. That final step can be taken by placing known, trustworthy CA's into the peergroup itself and delivering their public keys in root certificates with the Project JXTA platform.

4.2 Webs-of-Trust and the Poblano Algorithms

Poblano provides us with algorithms for calculating codat trust based on a peer's reputation in a given peergroup. Since a certificate is one form of codat, it follows that the Poblano algorithms can be applied to a peer's peergroup-keyring, i. e., a peer group member's collection of signed certificates for a given peergroup.

In sections 2 and 3 we showed how to compute CodatConfidence and PeerConfidence given the results of a keyword search. In the following discussion we assume that the keyword is "signed certificates," and that the only reponse that is permitted is the search target's peergroup-keyring's contents. Thus, in PeerGroup(i) we have the following two tables for matches to "signed certificates" where we have replaced CodatConf with CertConf:

PeerGroup(i)

CertConf Table

"signed certificates"

$CertConf_0$
$CertConf_1$
...
$CertConf_j$

PeerConf Table

"signed certificates"

$PeerConf_{cert0}$ $PeerConf_{rec0}$
$PeerConf_{cert1}$ $PeerConf_{rec1}$
...
$PeerConf_{certk}$ $PeerConf_{reck}$

The above CertConf table becomes the peer's keyring trust table for peerGroup(i), and each entry is associated with a signed certificate. The PeerConfidence values are user defined, and each peer's entry has two values. The first, $PeerConf_{cert,}$ measures a user's confidence in using given peer's certificate, i. e., public key, for securing a transaction. The second, $PeerConf_{rec,}$ rates that peer as a recommender, or certificate cosigner. The latter value measures the transitivity of trust. It answers the questions:

1) If Mary receives Sandi's certificate from Bob, and Mary does not know the subject, Sandi, of that certificate and Bob does, then is Mary willing to use Bob's recommendation of Sandi? Bob will have assigned a CertConfidence value to Sandi's certificate. If Mary uses this certificate as recommended by Bob, and to what degree, is determined by this value.

2) If Bob cosigns a certificate, how does Mary rate Bob's signature?

We think that trust may be transitive, how transitive can be explicitly, measured, and the degree of transitivity is up to the user. Other's have assummed otherwise [3]. If the $PeerConf_{rec}$ is less than 4, we say that the trust-relationship is **weakly-transitive**. It is from the local peer's perspective and is based on reputation.

CertConfidence is initally 4.0 for certificates originating on a peer, they are in fact certs, but becomes weighted by the **trust-path** and the $PeerConf_{rec}$ value if the source is not the issuer. In what follows we define how we calculate the CertConf Table's entries for trust-paths.

If we are using a Web-of-Trust, then if Mary's certificate is **self-signed**, Carl knows Mary, and he gets Mary's cert from Mary who is then the cert's subject, then using formula 2 for a path of length 1:

$$CertConf_{path} = \frac{PeerConf_{cert}(subject)}{4} \times CertConf_{provider}$$

The initial CertConf = 4.0, and PeerConf(subject) defaults to 2.0, or average, until set by the user.

Formula 7

Thus, for Mary we have:

$$CertConf_{path} = \frac{PeerConf_{cert}(Mary)}{4} \times 4.0$$

Thus, if Carl's peerConfidence in Mary is 3.0, the $CertConf_{path}$ is 3.0. This is Carl's confidence in Mary's certificate.

Next, if Chloe receives Mary's certificate from Carl, and Chloe's $peerConf_{rec}$ in Carl is 2.5, and Chloe does not know Mary, then we use:

$$CertConf_{path} = \frac{1}{4} \times \frac{(PeerConf_{rec} + PeerConf_{cert}(subject))}{2} \times CertConf_{provider}$$

The default $PeerConf_{rec}$ value is 1.0, or minimal. Relationships are initially

weakly-transitive.

Formula 8

Given the above, the $CertConf_{path}$ for Mary's certificate is:

$$CertConf_{path} = \frac{(2.5+2.0)}{8} \times 3.0 = 1.69$$

Here the CertConf for Mary's cert on Carl's system is 3.0 from the first example, and is used in the calculation in lieu of the default 4.0 value. Chloe rates Carl's recommendations at 3.0, and on her keyring Mary's certificate has a CertConf of 1.69.

A certifcate can have multiple signers as is done in PGP. If Mary's certificate is self-signed, and cosigned by Carl, and Chloe obtains the cosigned certificate from Carl, then the $certConf_{path}$ is as above, and equals 1.69. Chloe trusts Carl's CertConf in Mary.

On the other hand, if Carl cosigns the certificate, Chloe gets Mary's certificate from Mary, and does not know Mary, her $peerConf_{cert}$ in Mary is 2.0. Then, recalling that Chloe rates Carl's $PerConf_{rec}$ at 2.5, the $certConf_{path}$ is:

$$CertConf_{path} = \frac{(2.5+2.0)}{8} \times 4.0 = 2.25$$

But, if Chloe's $PeerConf_{cert}$ in Mary is 3.0, then we have:

$$CertConf_{path} = \frac{(2.5+3.0)}{8} \times 4.0 = 2.75$$

Chloe takes into account Carl as a cosigner. She certainly has the right to make Carl's $PeerConf_{rec}$ equal to 0, and not use transitivity of trust with respect Carl. In this case the above $CertConf_{path}$ will be 3.0.

The above can be applied to certificates with n signatures, n-1 cosigners, and the initial signer as P_n:

$$CertConf_{path} = \frac{1}{2 \times 4 \times n} \left(\sum_{i=1}^{n-1} PeerConf_{rec}(P_i) + n\,PeerConf_{cert}(P_n) \right) \times CertConf_{provider}$$

Formula 9

Finally, if a certificate is signed by a peerGroup CA, then that CA's root certificate is present on all peerGroup members machines. Such CA signed certificates will have a default $certConf_{provider}$ of 4.0, and the CA has default $peerConf_{cert}$ and $peerConf_{rec}$ of 4.0, thus giving all such certificates a local default $certConf_{path}$ of 4.0. Thus, for Mary we have the following for a cert received from a CA:

$$CertConf_{path} = \frac{(4.0 + 4.0)}{8} \times 4.0 = 4.0$$

A user may still apply formula 7 so that if Carl receives Mary's CA signed certificate from Mary, and $peerConf_{cert}$(Mary) is 3.0, then her certConf will be 3.0. This may effect Carl's willingness to do financial transactions with Mary, or sending her private mail using S/MIME [10] for example. These judgements are personal. And, downgrading such a certificate should be rare.

4.3 Updating the $PeerConf_{rec}$ values

It is possible that at any point in time the degree of transitivity of given peer's reputation as a recommender with respect to another peer is either too optimtistic or pessimistic. Thus, we need a way to measure, and correct our experience with respect to a peer's recommendations over time. We can do this because we have explicity defined $PeerConf_{rec}$ of each such recommender.

To this end, let K be the set of all CertConfs for which we have non-default values for both $PeerConf_{cert}$ and $PeerConf_{rec}$ for certificates uniquely recommended or cosigned by a given peer, P_0 . If K is empty, then we do not have sufficient experience to revaluate P_0. We then calculate the average recommendation for P_0 by defining:

$$coPeerConf_{rec}(P_0) = \frac{1}{|K|} \sum_{\alpha \in K} (CertConf_{path})_{\alpha}$$

where $|K|$ = number of certificates in K.

We can then calculate the direct-peerConf, i. e., as if we had gotten each certificate directly from the same subjects, i. e., $PeerConf_{rec}$ is set to 0:

$$direct\_peerConf = \frac{1}{|K|} \sum_{\alpha \in K} (PeerConf_{cert})_{\alpha}$$

The two values allow one to compare how the local peer's ratings correlate with the remote peer's, and permit the local peer to adjust his ratings accordingly if they don't agree.

For example: Suppose Mary obtains both Steve's and Chloe's certificates from Sally. If Mary gives Sally a $peerConf_{rec}$ value of 2.5, and the CertConf values of Steve and Chloe on Sally's system are 2.6 and 3.0 respectively. Then

$$coPeerConf(Sally) = (2.6 + 3.0)/2 = 2.8$$

If Mary rates Steve and Chloe with $peerConf_{cert}$ Values of 3.0 and 3.8, respectively, then by applying formula 7, we have:

$$direct\_PeerConf = (3.0 + 3.8)/2 = 3.4$$

Thus, Mary may be under rating Sally, and can adjust her $peerConf_{rec}$ value for her if she desires to do so.

5. Peer Identity and Authentication

For a JXTA peer to authenticate itself in a peerGroup a peer identity is required. Such an identity must be unique across the universe of peers. Also, certificates issued to a peer need a unique user identification (UUID). For X.509 [11] certificates this must be a X.500 distinguished name that is unique across the Internet. An example is

CN=Yu Chang, OU=Fun Water Slides, O=Summer Play, Inc., C=FR

PGP certificates require user information but are less stringent about the details. This can be "identity" information about the user such as her or his name, user ID, photograph, and so on [9]. In either case, we can generate a unique UUID. For example:

CN=UserName, OU=<Twenty digit pseudo-random id>, O=www.jxta.org, C=Country

A suitable concatenation of the above DN name identifiers is also suitable for a PGP certificate. Thus, given that each peer has its own certificate, self-signed, cosigned, or CA signed, we can create a peer identity by hashing the concatention of the UUID and the public-key fields, signing this hash with the private-key, and using the this digital signature as the identity. Since such a signature can be large, for large keys it is the key length, the first twenty bytes can be used as a digital finger print. Another possible finger print is the MD5 or SHA-1 hash of the private key. Both are reproducable only by the owner of the private key, and verifiable, and can be used as a challenge. The identity can be used as the peer's credential in JXTA messages.

Given a unique identity, a peer can use it as part of a peergroup's authentication policy which also may require a password to be created to grant, for example, group, account priviledges, and a renewal period. This is done over a private connection to protect the password. Finally, a group credential can be returned to the peergroup member which acknowledges and embodies the authorized priviledges. This same credential is then required whenever any of the associated peegroup services are used.

Admittedly, a great deal is missing in the above scenario, and the implication is that such a scheme may require JXTA authorization services. Peergroup members will need to be aware of which systems do authorization, and a source for list of URI's for these systems will need to be published using the JXTA advertisement mechanism.

Finally, while authorization services must support the JXTA protocols, their internals may not be open source for either proprietary or security reasons.

6. Mobile Credentials

Certainly, the ability to move one's private security environment from device to device is desirable. Having multiple identities, for example, is confusing and adds unwanted complexity to a security model. Since the private security enviroment can contain things like a user's private key, trusted root certificates, and peergroup credentials, mobility must be under the constraints of strongest security we can provide. If a private key is no longer private, one's security environment, and all of the associated relationships must be recreated from zero. As we mentioned earlier, there is a standards effort in the IETF's SACRED working group to accomplish the above.

"The SACRED Working Group is working on the standardization of a set of protocols for securely transferring credentials among devices. A general framework is being developed that will give an abstract definition of protocols which can meet the credential-transfer requirements. This framework will allow for the development of a set of protocols, which may vary from one another in some respects. Specific protocols that conform to the framework can then be developed [12]."

The problem is difficult and we see no point in duplicating this effort, and thus, will follow and perhaps contribute to the IETF work in this area. Those interested can begin by reading [12].

7. Implementation Notes

The purpose of this section is to provide some details related to the previous discussion that are important enough to state, but would have been distracting from the principal line of thought. Please bear this in mind as you read the following notes. Finally, many details are missing with respect to what a full security specification will provide. The final version of this paper should be such a specification.

7.1 Propagation of Keyword Search

In section 3 we discussed a three-step procedure for a given peer to follow in order to find codat associated with a given keyword. This procedure permits a keyword search to propagate from peer-to-peer until some peer finds the desired results, and receives the appropriate feedback.

As the search is propagated from $peer_i$ to $peer_{i+1}$, $peer_i$ sends his confidence in $peer_{i+1}$ to the peer that initiated the lookup (the peerID of the originator is contained in each successive query). This is necessary because if we permitted the intermediate peers to append their confidence in the successors this is a violation of the privacy of the predecessor and also permits the successor to falsify the recommendations.

The above forwarding process has a time-to-live which is currently 16. This limits the number of peers that can be visited to 16 for a given lookup as well as inadvertent search loops.

7.1.1 The Security Toolbox

Project JXTA has security API's and a library that has implements RSA, RC4, MD5, SHA-1, a psuedo-random number generator, MAC, and digital signatures. There are plans to add more ciphers, and Diffey-Hellman. A description of the existing API's and implementation can be found at:

http://security.jxta.org/Security-project.html

The default key strength is 512 bits for RSA, and 128 bits for RC4. These are there to fulfill U. S. export requirements for open source. The above toolkit is sufficient to begin our implementation.

7.2.1 Personal Security Environment, and codat privacy

Each user will have security parameters that must be keep absolutely private. Here one might find data such as the user's private key, root certificates, and peergroup credentials. In order to protect a user's personal security environment a passphrase will be required. This is user supplied and must be difficult to guess. For a key of

length 128 bits, and using english for the passphrase, there are 1.3 bits of information per character[5, page 174]. A secure passphrase should be 98 characters (128/1.3). MD5 can be applied to the hash phrase to produce a 16 byte key for the RC4 128 bit key block cipher for example. This can be used to encrypt the private data. Not many passphrases fulfill these information theory requirements for 128 bit keys. Clearly, shorter passphrases can give sufficient security, but the code must be aware of these limitations, given the chosen key size, and warn the user of passphrases of inadequate length.

One can then take the same passphrase and expand it by concatentating a few characters, e. g., "YY", then hash this and use the resulting key to create a MAC of the above encrypted data. This gives an integrity check for the private data, and prevents attacks on data that is encrypted by not MAC'd [13]. Using the same techniques and passphrase, we can securely protect local codat or remotely stored private codat.

7.2.2Keyrings

Each peer over time will acquire a local collection of certificates with their associated public keys. Such a collection is the user's keyring. Each peer will have at least one personal certificate.Thus, the keyring is non-empty. A peer should both publish the existence of this keyring, and distribute its contents on request using the JXTA protocols.

These protocols permit the creation of advertisements, for example, each peer has its peer own advertisement which contains static information describing that peer. The JXTA 1.0 peer advertisement has an XML tag reserved for security, and to add security to this version of JXTA we will advertise the peer's security pipeID in that XML field.

Each certificate on the peers keyring will have a reference which will contain at least the peerID, the DN of the certificate's subject or owner, and the local peer's CertConf for that certificate. This list of references defines a document, the peer's keyring-list, which will be accessible by the means of the peer's security pipe, and thus, publishes those keys which are exportable on that peer. One will be able to access any certificate given either its peerID or DN reference using the same pipe. We will be required to define a simple protocol for use on the security pipe.

The absence of a security pipeID in the peer advertisement implies that security services are not supported on that peer. Also, in future versions of JXTA the peer advertisement may not include the security pipeID since there is an ongoing project to reduce the size of these advertisements, and to make information like the security pipeID available on demand. In this case the security pipeID might be available through the PeerInfo protocol. We will argue that at least the availability of security services must be part of the peer advertisement.

7.2.3JXTA Transport Layer Security (TLS)

For private, peer-to-peer communication we are going to implement TLS within the constraints of the security model's trust spectrum discussed in the previous sections, and on top of the JXTA core protocols. We will use the TLS_RSA_WITH_RC4_128_SHA cipher suite from our Project JXTA security toolbox. And, rather than reinvent-the-wheel, we are considering the use of the Claymor System PureTLS code [14] to speed the implementation.

Again, our model implies that at the least secure end-point of this spectrum, self-signed certificates may be sent in the TLS handshake. And, thus, as has been previously discussed, the imposter-in-the-middle attack will be possible as it is for any PGP-like, Web-of-Trust where self-signing cannot prevent forged certificates. In section 4.1 we argued that cosigned certificates are more difficult to forge, and provide very good privacy.

Thus, we plan on implementing more than two points in the trust spectrum, i. e., self-signed, cosigned, and CA signed certificates. And with cosigning, and satellite CA's we can offer better than very good privacy TLS for zero-dollar cost.

7.2.4Peergroup Authentication

JXTA 1.0 has a framework for the Pluggable Authentication Modules (PAM) in place. Using peer identities as defined in section 5 we will add a peergroup authentication module to our PAM implementation. The initial authentication must be done by a peergroup member that has an authentication level of authority. It will return a peergroup credential containing at least the following fields:

1. Authorization privileges:
    1. data access: read, write
    2. authentication level: trial membership, full member, authority
2. Membership expiration date
3. Hash of member's password and the algorithm used.
4. Peer Identity of initiating authority
5. Digital Signature of the previous fields by initiating authority

The initial authentication will be done using TLS to keep the user's password private.

Further authentications to access other group members' systems must include the above credential, and thus can always be challenged by requesting the password, and reproducing the hash, after first verifying the credential with the public key of the initiating authority.

The above credential is neither complete nor fully tested. Rather, it is here to aid in arriving at a final specification. The preliminary goal is to put a JXTA peergoup, authentication infrastructure in place, and at the same time hammer out the details of the specification.

Reference

[1] B.T. Sniffen, "Trust Economies in the Free Haven Project," May 2000, http://theory.lcs.mit.edu/~cis/cis-theses.html.

[2] M. Waldman, A.D. Rubin and L.F. Cranor, "Publius: A Robust, Tamper-Evident, Censorship-Resistant Web Publishing System," Proc. 9th USENIX Security Symposium, pp 59-72, August, 2000

[3] A. Rahman, "A distributed trust model", http://www.acm.org/pubs/articles/proceedings/commsec/283699/p48-abdul-rahman/p48-abdul-rahman.pdf

[4] S.P. Marsh, "Trust in Distributed Artificial Intelligence", Artificial Social Systems, Springer Verlag, Lecture Notes in AI, Vol.830, pp94-112

[5] Bruce Schneier, "Applied Cryptography," John Wiley & Sons, Inc., 1996.

[6] A. Arsenaut, S. Farrell, "Securely Available Credentials - Requirements," draft-ietf-sacred-reqs-03.txt, June, 2001.

[7] D. Gustafson, M. Just, M. Nystrom, "Securely Available Credentials - Credential Server Framework," draft-ietf-sacred-framework-01.txt, March, 2001.

[8] T. Dierks, C. Allen, "The TLS Protocol, Version 1.0," rfc2246, January, 1999.

[9] "The International PGP Homepage," www.pgpi.org.

[10] B. Ramsdell, Editor, "S/MIME Version 3 Message Specification," rfc2633, June 1999.

[11] R. Housley, W. Ford, W. Polk, D. Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile," rfc2459, January, 1999.

[12] A. Arsenalut, S. Farrell, "Securely Available Credentials - Requirements," draft-ietf-sacred-reqs-03.txt, June, 2001.

[13] Steven M. Bellovin, "Problem Areas for the IP Security Protocols," in Proceedings of the Fifth Usenix UNIX Security Symposium, pp. 1-16, San Jose, CA, July 1996.

[14] Claymore Systems PureTLS, http://www.rtfm.com/puretls/.