Konfidi:[*] Trust Networks Using PGP and RDF

David Brondsema[†]
dave@brondsema.net

Andrew Schamp
schamp@gmail.com

ABSTRACT

Trust networks have great potential for improving the effectiveness of email filtering and many other processes concerned with the validity of identity and content. To explore this potential, we propose the Konfidi system. Konfidi uses PGP connections to determine authenticity, and topical trust connections described in RDF to compute inferred trust values. Between yourself and some person X whom you do not know, Konfidi works to find a path of cryptographic PGP signatures to assure the identity of X, and estimates a trust rating by an algorithm that operates along the trust paths that connect you to X. The trust paths are formed from public person-to-person trust ratings that are maintained by those individuals. We discuss the design of the network and system architecture and the current state of implementation.

Keywords

Semantic web, trust network, FOAF, RDF, OpenPGP, PGP, GPG, reputation, propagation, distributed, inference, delegation, social network

1. INTRODUCTION

As internet-based communication grows, it has experienced rapid growth of unscrupulous users taking advantage of the system to send spam and propagating viruses to users. This gives rise to two questions: How can one be sure that a message really comes from the indicated sender? How can one be sure that the sender can be trusted to send good messages?

There have been a number of attempts to answer either one question or the other. The OpenPGP encryption system [IETF, 1998] (hereafter PGP) has developed a web-of-trust which can help provide verification of an individual's identity; however, it does not allow the expression of any additional information about that individual's trustworthiness on matters other than personal identification. As for the second question, one answer that is growing in popularity is that of creating a network of trust between individuals who know one another and have good reason to trust their estimations of others. However, these systems can be subject to problems; suppose someone impersonating a trusted party provides incorrect data boosting the reputation of an untrustworthy party. A simple

---

[*]*Konfidi* is the Esperanto term for trust. A universal concept in a universal language seemed appropriate for what we hope will become a universal system.

[†]Both authors did the majority of this work as students at Calvin College.

rating system for reputation within certain domains, such as eBay online auctions, may be of some limited use. However, unless there is a system to verify the raters, they may also be susceptible to malicious users who manipulate ratings. Even if such systems can be guarded against such attacks, one should not have to base their trust in another person on ratings given by people that they neither know nor trust.

In this paper, we present a system that combines the a trust network with the PGP web-of-trust. We describe some difficulties in integrating the networks, and analyze various strategies for overcoming them. We then describe our structure for representing trust data, and our methods for making trust inferences on this data. Finally, we discuss the our proof-of-concept software for putting this trust to use.

2. RELATED WORK

We have incorporated into our project a number of existing technologies designed to serve various purposes. We introduce them here, and explain later in the paper how we have integrated them. We also include a discussion of related academic research on the relevant topics.

2.1 Representing Trust Relationships

There seems to be a general lack of psychological research on ways of representing trust relationships between individuals and procedures for inferring unspecified trust values. We found no recommendations for a particular scheme for modeling trust relationships or networks mathematically. Most work on this topic in the fields of mathematics and computer science adopts an arbitrary model appropriate to the algorithm under consideration. Guha points out [Guha *et al.*, 2004] that there are compelling reasons for a trust representation scheme to express explicit distrust as well as trust.

2.2 Trust Networks and Inferences

There are several different propagation strategies for weighted, directed graphs [Richardson *et al.*, 2003] [Abdul-Rahman & Hailes, 1999] [Guha *et al.*, 2004]. For the most part, however, the work is concerned with mathematical description of the networks and their operations, and do not have much in the way of practical application. While these issues are of interest and relevance, they concern only the subsystem and do not discuss the design of a larger infrastructure.

Jennifer Golbeck, at the University of Maryland, is doing work on trust systems [Golbeck, 2005a] that is similar to our work on this project. Like us, she uses a Resource Description Framework (RDF) [W3C, 2005a] schema with the Friend of a Friend (FOAF) [Brickley, 2005a] RDF schema to represent trust relationships and

a rating system[1]. She has created TrustMail [Golbeck, 2005b], a modified email client that uses her trust network. She is more concerned with an academic approach than a pragmatic one, since this field is still growing rapidly and she emphasizes her research on other applications and implications of semantic social networks.

Golbeck suggests an important distinction between belief in statements and trust in people [Golbeck & Hendler, 2004]. While networks of both kinds can be created, the latter are usually smaller and more connected. Golbeck argues that in a combined network of trust in people and of belief in statements, a path composed of trust edges and terminating with a belief edge is equivalent to, and on average smaller than, one composed entirely of belief edges. Thus, a trust network comprising mostly trust edges allows for simpler traversal.

2.3 The Semantic Web

In addition to Golbeck, a number of others have explored the usefulness and implications of expressing trust relationships in the Semantic Web.

The FOAF project is an RDF vocabulary that can be used to represent personal data and interpersonal relationships for the Semantic Web. Users create RDF files describing `Person`[2] objects which can specify name, email address, and so on, but more importantly, they can express relationships between `Person` objects. There are a number of tools in development for processing FOAF data and traversing references between FOAF RDF files. These tools can aggregate information because RDF often uses uniform resource indicators (URIs) to identify each individual object.

Dan Brickley has made a practical attempt to investigate the use of FOAF, particularly the `mbox_sha1` property, to automatically generate email whitelists. By hashing the sender's email address using SHA1, privacy is protected (and the address cannot be gathered by spiders), and so users can share whitelists of `mbox_sha1s` of addresses they know not to send spam. Then for all incoming mail, the sender's address is hashed and the whitelist searched for the resulting value, and then is filtered accordingly. This use of FOAF is promising, but since it is decentralized, it is difficult for updates to propagate [Brickley, 2005b]. No effort is taken in this project to verify the sender's identity.

2.4 Email Filtering

Filtering email to reduce unsolicited email has received considerable attention in many areas. Domain-level solutions, such as Sender Policy Framework (SPF) [Wong, 2004] and DomainKeys Identified Mail (DKIM) [DKIM, 2005], are designed mostly to prevent phishing (emails with a forged From: address to trick users into divulging personal information) and also assume that a domain's administrator can control and monitor all its user's activities. Greylisting and blacklisting often have too many false positives and false negatives. User-level filtering, which Konfidi does in the context of email, is not very common. Challenge-response mechanisms to build a whitelist are tedious for the sender and receiver and do not validate authenticity. Content-level testing is the most common, but Bayesian filtering and other header checks are reactionary and must be updated often, and are becoming less effective as spammers create emails that look ever more legitimate, attempting either to fool the filter or to distort the probabilities.

There has been some work to bring authentication to email through the domain-level efforts of SPF and DKIM. Their goal is to prevent phishing by assuring authenticity through cryptographic data in DNS records. These approaches limit their applicability to domain-related data such as email or webpages and do not address any issues of trust, since DNS records must be assumed to be authentic. Also, the granularity of the system is too coarse: cryptographic keys are normally created on a per-domain, not per-address, basis.

2.4.1 Trust Inference Using Headers

Boykin and Roychowdhury discuss ways to infer a relationship based on existing data [Boykin & Roychowdhury, 2004]. They suggest scanning the `From:`, `To:` and `Cc:` headers and building a whitelisting database based on relationships indicated by the recipients. This seems to work fairly well, but there is often not enough data to make the spam/not-spam decision because it is based only on the user's own previously received messages. They clearly state a cryptographic solution would be ideal to verify the sender's identity.

2.4.2 Trust Inference Using PGP

One approach would be for a Mail User Agent (MUA) to find a path from any PGP-signed email's sender to the recipient.[3] There are some MUA plugins, such as Enigmail [Brunschwig & Saravanan, 2005], that implement some of this. Enigmail uses PGP to sign emails and validate any emails that are received with a PGP signature, fetching keys from the keyserver when necessary. If there is a short enough path of signatures from the recipient to the sender, the signature is considered "trusted". It does not fetch keys in an attempt to find such a path; you must already have the keys locally that form the path. Fetching all the keys along the path would be necessary, but is problematic for reasons explained later.

Using this approach to filter spam would require that most users digitally sign email messages, and it depends on users to be aware of known spammers and avoid signing their keys. However, the recommended PGP keysigning practices require only the careful verification of the key-holder's identity, and a signed key does not entail anything about trustworthiness in other areas. Furthermore, if the identification requirements for keysigning are met, even by a spammer, it would be unfair to refrain from signing that spammer's key[4]. Whether a user should be trusted to send good email, and not spam, is information over and above that expressed in the PGP web-of-trust itself, so another system would be required to encode such information.

Another serious flaw in this approach is this: because key signatures are listed with the signed key and not the signing key, the MUA must search for a path between users that can only be constructed from the sender to the recipient. Since these paths would have to be built starting from the sender, a spammer or other malicious user could generate a large number of fake keys that are inter-signed, and then use these keys to sign their sender's key. This could inundate the client's search domain making such a search impractical. A deluge of false information would put undue strain on the clients and keyserver infrastructure, and would amount to a denial-of-service, of sorts. Existing keyserver infrastructure provides no effecient way to tell which keys a particular key has signed, which would allow searches in the reverse direction that are not susceptible to this misuse.

2.5 PGP Web of Trust

---

[1]Though both our ontologies and ratings are different in significant ways, which we will address later.
[2]According to RDF standards, the names of objects are capitalized, while the names of properties remain lowercase.

[3]In the web-of-trust, nodes are PGP keys and edges are key signatures. Paths are made when the recipient has signed someone's key, who has signed another key, and so on all the way until a signature is found on someone who has signed the sender's key
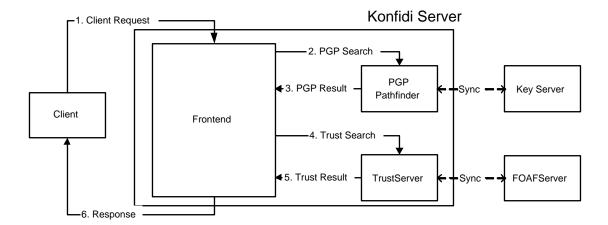[4]In fact, such positive identification might be of use.

Figure 1: Konfidi Architecture

Wotsap [Cederlöf, 2005] is a tool to work with the PGP web-of-trust. From a keyserver it creates a data file with the names, email addresses, and signature connections of all keys from the largest strongly connected set of keys, but no cryptographic data. For technical reasons, it does not include all keys or even all reachable keys. Wotsap includes a python script to use this data file to find paths between keys and generate statistics.

2.6 Summary

This related work forms many of the building blocks, both technical and theoretical, for our work. A proper system should determine authenticity through a decentralized network and determine trust in a topic through a similar network topology. We integrate PGP, RDF and FOAF, and design ideas from Golbeck, Guha, and others. We are extending FOAF with an RDF trust ontology to represent our trust network, which ties into the PGP web-of-trust to verify authorship and identity. We expanded Golbeck's trust ontology to a relationship-centered model with values in a continuous range which represent trust and distrust.

3. KONFIDI

Konfidi refers to the trust network design, the ontology used to encode it, and the software to make it usable. The central idea is that between yourself and person X whom you do not know, there is a path of PGP signatures to assure the identity of X. An estimated trust rating can then be computed by some algorithm that operates along the trust paths that connect you to X. Figure 1 shows the components of the Konfidi architecture and how they relate to external components and one another. The numbered paths indicate the steps in the process:

1. A client makes a request to the Konfidi server, indicating the source and the sink.[5]
2. The frontend passes the request to the PGP Pathfinder, which verifies that some path exists from the source to the sink in the PGP web-of-trust.
3. The Pathfinder returns its response.
4. If thre is a valid PGP web-of-trust connection, the frontend passes the request to the TrustServer, which traverses the

Konfidi trust network that is built from data kept up-to-date by the FOAFServer.
5. The TrustServer responds with the inferred trust value or an appropriate error message.
6. The Frontend combines the responses of the Pathfinder and the TrustServer, and sends them back to the client.

In the remainder of this section, we discuss the underlying data structure for representing trust, how it is implemented in these steps, and the rationale for the system design.

3.1 Trust Ontology

In the current research on trust inference networks, there seem to be two general kinds of representations: one that uses discrete values for varying levels of trust, and one which uses a continuous range of trust values. Both return an answer in the same range as their domain. Either kind of representation could be roughly mapped onto the other, however, a continuous range would allow more finely-grained control over the data. Further, the inferred trust values returned by searches would not have to be rounded to a discrete level, which would lose precision.

In our representation, trust is considered as a continuum of both trust and distrust, not a measure of just one or the other. For example, if Alice trusts Bob at some moderate level (say, .75 of a scale of 0 to 1), then it seems that she also *distrusts* him at some minimal level (say, .25). If Alice trusts Bob neutrally, then she trusts him about as much as she distrusts him. If she distrusts him completely, then she doesn't trust him at all. But in all of these cases, there is a trade-off between trust and distrust. Only in the extreme cases are either of them eliminated completely. Our trust model represents a range of values from 0 to 1, treating 0 as complete distrust, 1 as complete trust, and 0.5 as neutral. This also makes many propagation algorithms simpler, as we'll discuss later.[6]

3.1.1 Distrust

The choice of representation is closely related to the concern that it an account of distrust. If the trust network contained values ranging from neutral trust to complete trust, then everyone in the network is trusted, explicitly or by inference, on some level at

---

[5]Source is defined as the entity at the beginning of a desired path, and usually the one making the request. Sink is defined as the entity to which the path leads

[6]Considering trust in this range naturally evokes the possibility of applying probability theory, however, such approaches are beyond the scope of this paper. Further consideration is merited, and might be implemented strategically as discussed in Section 3.2.3.

or above neutral. If the system makes a trust inference between Alice and Bob at one level, but Alice really trusts Bob at a different level, she can explicitly state this previously implicit trust to have a more accurate result (for herself and for others who build inference paths through her to Bob). But, suppose that Alice feels strong negative feelings about Bob. In this case, she would still only be able to represent this relationship as one of neutral trust. So, the trust network must account for distrust in some reasonable way.
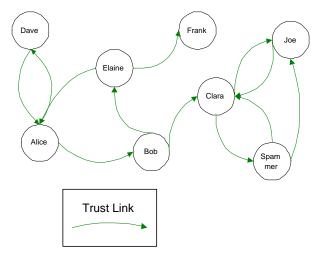

Figure 2: An Example Trust Network

One of the difficulties of using explicit distrust in an inference network is that it is unclear how inferences should proceed once a link of distrust has been encountered. Consider a trust network like that depicted in Figure 2. Suppose Alice distrusts Bob, and Bob distrusts Clara. As Guha points out [Guha *et al.*, 2004], there are at least two possible interpretations of this situation. On the one hand, Alice might think something like "the enemy of my enemy is my friend" and so decide to put trust in Clara. On the other hand, she might realize that if someone as scheming as Bob distrusts Clara, then Clara must *really* be an unreliable character, and so decide to distrust her. Further, suppose Bob expressed trust for Elaine. At first consideration, it might seem reasonable to simply distrust everyone that Bob distrusts, including Elaine. But suppose there were another path through different nodes indicating some minimal level of trust for Elaine. Which path should be chosen as one which provides the correct inference? Since Konfidi represents trust on an interval, and concatenates (combines trust path ratings) values by multiplication, any distrust will make the computed score drop quickly below the minimum threshold. This effectively stops propagation along a path when distrust is encountered.

3.1.2 Data Structure

Golbeck's ontology represents trust as a relationship between a person and a composite object comprising a topic, a person, and a rating[7]. However, this representation requires trust relationships to be in the context of a person. Accordingly, it may be difficult to associate additional information with the trust relationship.

In our schema, we represent each trust relationship as an object, and the trusting person and the trusted entity (typically a person) are associated with that object. Each relationship goes one-way from truster to trusted, but since the truster is responsible for the accuracy

---

[7] `Subject`, trusted `Person`, and `Value` according to her terminology

of the information, that avoids the pitfalls of the PGP web-of-trust implementation as discussed in Section 2.4.2. Trust relationships also have trust items specified. See Section 3.1.4 for a specific description of the structure.

Because the trust relationship is represented as its own object, other attributes may be added as the need arises, such as the dates the relationship began, annotations, etc.

3.1.3 Trust Topics

If other attributes about a trust relationship could be expressed, in addition to the rating values, then a system like Konfidi would be useful in many wider scopes than email spam prevention. To describe this, an attribute of trust topic is used. A natural feature of interpersonal trust relationships is that there can be many different aspects of the same trust relationship.

For example, suppose Bob is a master chef, but is terribly gullible about the weather forecast. Alice, of course, knows this, and so wants to express that she trusts Bob very highly when he gives advice for making souffle, but she does not trust him at all when he volunteers information about the likelihood of the next tornado. Suppose she only knows Bob in these two capacities. Any trust inference system should not average the two trust values and get a somewhat neutral rating for Bob, for that would lose important information about each of those two trust ratings, the only information that made these ratings useful in the first place.

Suppose also that, given only the above trust ratings, the system tried to make an inference on a subject that was not specified. Perhaps Alice has some general level of trust for Bob that should be used when there is no specific rating for the topic in question. See the discussion in Future Work for our proposal for a hierarchical system of topics that might account for this situation. As the number of topics rises, the amount of information stored increases in size. However, since trust topics and values are attributes of the trust relationship, they need not be represented as additional edges in the graph, they can be stored as additional information attached to existing edges.

3.1.4 OWL Schema

As the FOAF project grows in popularity, an infrastructure is growing to support it, as mentioned in Section 2.3. Like FOAF, Konfidi also uses RDF to represent trust relationships, so that it can take advantage of the infrastructure, and since the specification of trust relationships fits in naturally alongside existing FOAF properties. In addition to the FOAF vocabulary, there is a vocabulary called WOT which describes web-of-trust resources such as key fingerprints, signing, and assurance [Brickley, 2005c]. Because Konfidi's vocabulary makes use of FOAF and WOT vocabulary elements, then it can take advantage of the established standards and make the extensions compatible with existing FOAF-enabled tools.

Konfidi uses the Web Ontology Language (OWL) [W3C, 2005b] to define the RDF elements that make up the Konfidi trust ontology. OWL builds on the existing RDF specification by providing a vocabulary to describe properties and classes, and their relations. The Konfidi trust ontology provides two objects and five properties, which, in conjunction with the existing FOAF and WOT vocabularies, are sufficient to describe the trust relationships that Konfidi requires.

The primary element is `Relationship`, which represents a relationship of trust that holds between two persons. There are two properties that are required for every `Relationship`, `truster` and `trusted`, which indicate the two parties to the relationship. Both `truster` and `trusted` have `foaf:Person` objects as their targets. These `Person` objects should also contain at least

one `wot:fingerprint` property specifying the PGP fingerprint of a public key held by the individual the `Person` describes. This property is required for verification; if no `fingerprint` is available, then Konfidi cannot use the relationship. In general, any object described in RDF with a resource URI can be the `trusted` party, such as specific documents or websites, but for simplicity in our examples, we will focus on persons. which may be defined in the same file, inline, or in external documents indicated by their resource URIs. Because it does not matter where the `foaf:Person` data is stored, users may keep files indicating trust relationships separate from main FOAF files. However, to ensure authenticity, any file containing one or more `Relationship` objects must have a valid PGP signature from a public key corresponding to the `fingerprint` of each `Person` listed as a `truster` in that file. As described in Section 4, flexibility in data location can have a number of advantages.

In addition to `truster` and `trusted`, each `Relationship` requires at least one `about` property, which relates the trust `Relationship` to a trust `Item`. A `Relationship` is not limited in the other properties it can have, so the schema can be extended to include auxiliary information about the relationship, such as when it began, who introduced it and so on without having an effect on the requirements of Konfidi. Each `Item` has two properties belonging to it. The `topic` property specifies the subject of the trust according to a trust topic hierarchy[8] and the `rating` property indicates the value, according to the 0-1 scale of trust (specified in Section 3.1.2) that is assigned to the relationship on that topic.

A `Relationship` may have more than one `Item` that it is about. For example, remember the example given above, in which Alice trusts Bob highly about cooking, and distrusts him somewhat about the weather. This might be represented in our ontology as something like the following[9]:

```
<Relationship>
  <truster rdf:resource="#alice123" />
  <trusted rdf:resource="#bob1812" />
  <about>
    <Item>
      <rating>.95</rating>
      <topic rdf:resource="#cooking" />
    </Item>
  </about>
  <about>
    <Item>
      <rating>.35</rating>
      <topic rdf:resource="#weather" />
    </Item>
  </about>
</Relationship>
```

For RDF corresponding to some of the network depicted in Figure 2, see Appendix B. See Appendix A for the full OWL source code of the schema.

3.2 The Konfidi Server

The Konfidi server handles requests for trust ratings, verifies that a PGP connection exists, and traverses the internal representation to find a path. Since these three tasks are so distinct, all of Konfidi is divided into three parts. Figure 1 shows the relationships between a frontend which listens for requests and dispatches them, and two internal components, one to search the PGP web-of-trust and another to query against Konfidi's trust network. This separation, in addition to simplifying the design by encapsulating the different functions, also allows for increased flexibility and scalability. Each part is loosely coupled to the other parts, with a simple API for handling communications between them.

3.2.1 Frontend

Like the FOAFServer described in Section 4, the TrustServer's frontend is a web service, using the REST architecture to receiving and answering queries. It runs on the Apache web server, using the mod_python framework. Queries are passed in using HTTP's GET method, and responses are returned in XML, which a client application may parse to retrieve the desired data.

When a query is received, the Frontend passes the source and sink fingerprints to the PGP Pathfinder, and, if a valid path is found, to the TrustServer[10]. The Frontend then builds the response document to return to the client. The client may, for simplicity, request only the trust rating value instead of the full XML document.

3.2.2 PGP Pathfinder

As mentioned in Section 2.4.2, the PGP web-of-trust is not sufficient in itself for determining trust. However, it is necessary for the proper operation of Konfidi because it is required to verify the identity of the sink. Verifying that the document's signing key matches the key of the sink in the Konfidi trust network ensures that when Konfidi finds a topical trust inference path from source to the sink, it is valid. If the author of a document were not identified correctly, someone might forge the trust data, and Konfidi would return an incorrect result.

The Konfidi trust network is not coupled to the PGP web-of-trust for two reasons. First, the set of people one might wish to indicate trust for in Konfidi will likely not be the same as the set of those whose keys you are able to sign. For example, a researcher in Sydney may work closely with another in Oslo, and so trust that person's opinion highly in matters relating to their research. But it may be some time before they are able to meet in person to sign each other's keys directly. However, a valid path in the PGP web-of-trust may already exist connecting them.

Second, requiring users to sign the key of each person they want to add to their Konfidi trust networks adds additional difficulty which should otherwise be avoided. In keeping with the recommended practices for PGP, two individuals must meet in person and verify photo identification before they are to sign each other's keys. If this had to be done every time a Konfidi trust link were added, the extra hassle might entice users to grow lax in their keysigning policy, failing to properly complete such requirements. This attitude, when widespread would substantially weaken the web-of-trust. By keeping the PGP web-of-trust separate from the Konfidi trust network, the strength of the web-of-trust will not be weakened needlessly.

Usability becomes an additional advantage of separating the two trust networks. Aunt Sally can still use Konfidi to indicate trust if she and only one other person, say, a more technically savvy nephew, sign each other's keys. She will then be connected to the PGP web-of-trust within a reasonable distance of other family members which she is likely to include in her trust network. Now there is no need to teach Aunt Sally the requirements for key

---

[8]yet to be developed

[9]That is, supposing that the objects `alice123` and `bob1812` are defined elsewhere in the same file, and `cooking`, and `weather` are defined as part of the topic hierarchy.

[10]Strictly speaking, either query is optional. The PGP backend may be skipped to run tests on large sets of sample data, and the trust backend may be skipped if the system is to be used as an interface to the PGP web-of-trust only.

signing, and explaining why they must be done for each person she wishes to add to her Konfidi trust network. The system is easier to use, and the web-of-trust is less likely to be compromised[11].

The frontend uses drivers in a Strategy pattern [Gamma *et al.*, 1995], so that different subsystems for doing PGP pathfinding can be interchanged as they are developed. The current version utilizes the Wotsap pathfinder [Cederlöf, 2005] described in Section 2.5.

3.2.3 TrustServer

The Konfidi trust backend is responsible for storing the internal representation of the Konfidi trust network, incorporating updates into the network, and responding to queries about the nodes in the network.

The TrustServer can register with a FOAFServer as a mirror to receive notification whenever a FOAF record with trust information is added or altered. This can also allow it to synchronize with the FOAFServer after a period of down time in which new records have been added. The TrustServer currently assumes that the FOAFServer has verified the signatures of the FOAF records it stores, freeing it from the computational burden of fetching the signing keys and verifying the signature. See Section 4 for more explanation of the FOAFServer and its functions.

When it updates a record, the TrustServer parses the RDF input data and adds the relevant information to its internal representation of the trust network, which is a list of all `foaf:Person` records indexed by fingerprint and links to each `Person` marked as trusted, along with topic and rating data. The updated data will then be available for subsequent queries. This scheme accomplishes the goal of having trust links available in the proper direction, from source to sink, and avoiding one species of bogus data attack, as discussed in Section 2.4.2.

Let m be the number of persons, n the number of trust edges, l the average length of a path between two persons, k the average number of topics per relationship, o the number of persons being updated, and p the number of edges being updated. This representation requires $O((m + n) * k)$ space to store and on average, $O(m * l)$ time to search, and $O(o + p)$ time to update. On the other hand, a representation of a completely solved network, storing the trust values between any two individuals, requires $O(m^2 * k)$ space, but makes trust queries take a maximum of $O(1)$ time. However, such a representation requires $O(m^2 * l * k)$ time to solve, which it must do again after every update, since it must recompute the value for every pair.

The tradeoff between storage space and query time makes it hard to settle on a representation. Perhaps a compromise between a "live" system that incorporates incremental updates with slow queries, and a system that updates its network several times a day, rather than on each update, could provide better performance. Most users will not need up-to-date links with every user, since their queries will most likely be over a rather limited subset of the network. Caching of previously computed trust values on the user's end, with periodic updating, might also make a difference.

It may also be advantageous to store trust links going the other direction, perhaps for local representation analysis, or auxiliary information like name or email address. Other information, such as when the record was last updated, could allow for record caching that might improve performance.

Because of the apparent lack of psychological research on trust representations, we have again implemented the Strategy pattern

---

[11]While the effects of individual keys being compromised on the web-of-trust as a whole would be restricted to the key's neighborhood in the web, as this happened with greater frequency, the usefulness of the entire web would be undermined.

[Gamma *et al.*, 1995], for the trust propagation algorithm. This allows additional propagation strategies to be used as they are developed. The algorithm we present is the one that seemed most intuitive to us; we expect there are ones that more accurately reflect the human understanding of trust. It does simple multiplicative propagation over each link in a path. It uses a breadth-first search, prioritized to follow whichever path has highest value after each iteration, to find the shortest path between source and sink, if one exists:

```
function findRating(source, sink):
  keep a priority queue of all paths
  until the sink is found
    find the path with the highest rating
    find the link not already seen
      concatenate ratings from path and link
      add the path and rating to the queue
  return the path rating
```

The concatenation algorithm used simply multiplies trust ratings along each step in the path, with a fall-off of $x^{1/2}$ to keep the ratings from falling too quickly:

$$r = \prod_{i=0}^{n-1} Rating(i, i+1)^{1/2}$$

where $Rating$ returns the rating on the edge of two adjacent nodes.

Figure 3 shows an example of how the PGP web-of-trust and the Konfidi trust network might be combined. According to the algorithm, Dave's inferred trust of Clara on the topic of email is $0.8^{1/2} * 0.9^{1/2} * 0.7^{1/2} = 0.71$.

Note that while most PGP edges are two way, the usual outcome from a keysigning event, trust edges are more likely to be one way only. The trust edges are labeled to indicate trust rating and topic, to show how a certain path through the network could yield a low rating for the spammer. The RDF data of this labeled network can be found in Appendix B.
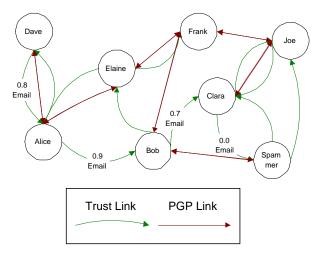

Figure 3: Combined Trust Network

4.   FOAFSERVER

The Konfidi server uses data from PGP keyservers to act on identity trust. To act on topical trust, we need a similar data store. This is not necessarily within the scope of Konfidi, but is a necessary prerequisite. We created the FOAFServer to fulfill this need.

The FOAFServer is a web service that stores and serves FOAF files that include trust relationships as specified by our trust ontology. A separate FOAF file is stored for each person, identified by their PGP fingerprint. All FOAF files must be PGP signed by the owner to prevent false data from being submitted and to prevent unauthorized modification of someone else's data. When a FOAF file is requested, the PGP signature is included so that it may be verified by a client.

Multiple FOAFServers will be available for public use and will synchronize their contents. Like the SKS PGP Keyserver[Minsky, 2004], anti-entropy reconciliation will be used, in which, at each time of synchronization, servers synchronize the entire database regardless of the current states. There is a trade-off between computation and communication expenses. This is preferred to the rumor-mongering reconciliation used by traditional PGP keyservers, in which only the most recent updates are pushed to other servers, since this does not allow servers to be out of communication for an extended period of time. Synchronization data will be PGP signed to maintain trusted secure communication channels everywhere.

Since the primary function of the FOAFServer is data storage, it may hold FOAF files that are not related to trust. A FOAF server may be configurable to act as one that is used for trust relationships, pet information, or résumés. Moreover, RDF features a `seeAlso` tag so a single FOAF file hosted on a FOAF server may refer to more FOAF data hosted elsewhere. This gives the owner flexibility, including encrypting or limiting access to a FOAF file hosted under his or her direct control.

Our FOAFServer is built with the Apache HTTP Server and mod_python using principles of REST architecture. Various clients can retrieve and set data using HTTP `PUT` and `GET` methods on URIs like `http://domain.org/foafserver/9BB3CE70`. `PUT` requests must be `Content-Type:multipart/signed` and `GET` requests are served with a content appropriate to the request's `Accept:` header. A web form for uploading FOAF files and their signatures is also provided.

Synchronization has not been implemented yet. Currently the TrustServer listens on a port for filenames that it should load into its memory. When someone updates a file via the FOAFServer, it sends the filename to the TrustServer update listening port so the TrustServer reloads it. Thus currently the FOAFServer and Trust-Server must run on systems with access to the same filesystem.

5. CLIENTS

The PGP, FOAF, and Konfidi servers each have clients which end-users use to view and modify the data.

5.1 PGP Clients

Many clients have already been written to interact with PGP keyservers with the Horowitz Key Protocol (HKP), a standard, yet undocumented[12], set of filenames and conventions using HTTP. The server itself also provides web forms to search for and view keys. It may be useful to integrate a PGP client with other Konfidi clients to provide a more cohesive user interface to the system.

Many MUAs have plugins or extensions to send `multipart/-signed` PGP emails. Users should use these for Konfidi to be useful for email filtering.

5.2 FOAF Clients

The FOAFServer provides some web forms to allow users to upload FOAF documents and PGP signatures. We plan to develop desktop software for users to create, sign, and upload their FOAF documents. See Section 4 for a summary of the FOAFServer HTTP interface.

5.3 Konfidi Clients

Only the Command Line Email Client has been written yet, but most clients will work similarly, depending on the context in which they are used. We expect that to make Konfidi widely popular as a method of stopping spam, a plugin or extension for every major MUA will need to be written.

5.3.1 Command Line Email Client

This client is designed to be invoked from a mail processing daemon, such as procmail [Guenther & van den Berg, 2001]. It reads a single email message from standard in, adds several headers, and writes the message back to standard out. By doing this, a MUA can filter the message based on the value of the added headers.

The client does the following tasks:

1. determines the source's PGP fingerprint (normally from a configuration file)
2. removes any existing X-Konfidi-* and X-PGP-* headers[13]
3. stops, if the message is not multipart/signed using PGP
4. stops, if the PGP signature does not validate
5. stops, if the `From:` header is not one of the email addresses listed on the key used to create the signature
6. queries the Konfidi server with the topic "email" and the fingerprints of the source (recipient) and sink (signing party)
7. receives the computed trust value from the Konfidi server

The client adds the following headers to the email:

Header	Value
`X-PGP-Signature:`	valid, invalid, etc
`X-PGP-Fingerprint:`	the hexadecimal value
`X-Konfidi-Email-Rating:`	decimal in [0-1]
`X-Konfidi-Email-Level:`	*s for easy matching e.g., `-Level:  *******`
`X-Konfidi-Client:`	`cli-filter 0.1`

If the client stops at any point, it will still add appropriate headers before writing the message to standard out.

6. FUTURE WORK

There are a number of things to be done to develop Konfidi from a proof-of-concept to a useful system.[14] As we've mentioned above, one thing we need most is a good base of psychological and sociological research backing up our trust representation and propagation, or suggesting a new one. Unfortunately, we must leave this to the experts in psychology. The rest of the system can be developed in its absence, so long as it is understood that we have just approximated how trust might work.

As we've said, a trust system is only as useful as it is trusted. Thus, a system of secure communication between every different component is required, most likely using PGP multipart/signed data. It is hard to say how a user's trust in a system like Konfidi can be represented within itself, but that may have implications, too.

In addition to plugins at the level of the user's MUA, Konfidi could be incorporated into the email infrastructure at the Mail Transfer Agent (MTA) level. Thus, a system could check Konfidi and add query results to every email message that it delivers to the user.

---

[12]Expired `Internet-Draft` draft-shaw-openpgp-hkp-00.txt does document the protocol

[13]This is done in case a spammer sends an email with invalid headers in an attempt to get past the filter.

[14]Development is ongoing at `http://www.konfidi.org/`

As the scope of Konfidi naturally expands to include things other than email, other clients will be developed. One possible client is a web browser extension to query pages when they are visited. This would work with server extensions that allows PGP signatures to be associated with webpages and served as `multipart/signed`.

For trust topics to be really useful, some sort of hierarchy is in order. Topics ought to standardized so that it is clear in what circumstances they apply, and how they relate to one another. So, for example, if Alice trusts Bob about internet communication in general, then if a query is made about email (a descendant of internet communication) and no explicit email rating is given, then Konfidi traverses up the hierarchy until some more general trust rating is found, and applies that.

7. CONCLUSIONS

With further research into psychological models of trust and social implications of widespread accountability, Konfidi promises to be a useful tool to bring distant trusted subjects into one's own realm of trusted subjects. Significant work remains to be done with Konfidi, even to apply it to email communication, but we believe it is a desirable and necessary system in a globalizing society.

8. ACKNOWLEDGMENTS

We would like to thank Keith Vander Linden for advising us on this project and giving feedback on drafts of this paper, and Earl Fife, Jeremy Frens and Harry Plantinga for their advice on specific matters.

References

Abdul-Rahman, Alfarez, & Hailes, Stephen. 1999. Relying On Trust To Find Reliable Information. *In: Proceedings 1999 Internation Symposium on Database, Web and Cooperative Systems (DWACOS'99)*.

Boykin, P. Oscar, & Roychowdhury, Vwani. 2004. *Personal Email Networks: An Effective Anti-Spam Tool*. http://www.arxiv.org/abs/cond-mat/0402142.

Brickley, Dan. 2005a. *friend of a friend (foaf) project*. http://www.foaf-project.org/.

Brickley, Dan. 2005b. *RDF for mail filtering: FOAF whitelists*. http://www.w3.org/2001/12/rubyrdf/util/foafwhite/intro.html.

Brickley, Dan. 2005c. *WOT RDF Vocabulary*. http://xmlns.com/wot/0.1/.

Brunschwig, Patrick, & Saravanan, R. 2005. *Enigmail Website*. http://enigmail.mozdev.org/.

Cederlöf, Jörgen. 2005. *Wotsap: Web of Trust Statistics and Pathfinder*. http://www.lysator.liu.se/~jc/wotsap/.

DKIM. 2005. *DKIM Website*. http://mipassoc.org/dkim/.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Golbeck, Jennifer. 2005a. *Computing and Applying Trust in Web-based Social Networks*. University of Maryland. http://trust.mindswap.org/papers/GolbeckDissertation.pdf.

Golbeck, Jennifer. 2005b. *TrustMail*. http://trust.mindswap.com/trustMail.shtml.

Golbeck, Jennifer, & Hendler, James A. 2004. Accuracy of Metrics for Inferring Trust and Reputation in Semantic Web-Based Social Networks. *Pages 116–131 of: Engineering Knowledge in the Age of the Semantic Web, 14th Interational Conference, Proceedings*.

Guenther, Philip, & van den Berg, Stephen R. 2001. *Procmail Website*. http://www.procmail.org.

Guha, R., Kumar, Ravi, Raghaven, Prabhakar, & Tomkins, Andrew. 2004. Propagation of Trust and Distrust. *Pages 403–412 of: Proceedings of WWW 04*ACM, for ACM.

IETF. 1998. *OpenPGP Message Format*. http://www.ietf.org/rfc/rfc2440.txt.

Minsky, Yaron. 2004. *SKS Keyserver*. http://www.nongnu.org/sks/.

Richardson, M., Agrawal, R., & Domingos, P. 2003. Trust Management for the Semantic Web. *Pages 351–368 of: Proceedings of the Second International Semantic Web Conference*.

W3C. 2005a. *Resource Description Framework (RDF)*. http://www.w3.org/RDF/.

W3C. 2005b. *Web Ontology Language (OWL)*. http://www.w3.org/2004/OWL/.

Wong, Meng Weng. 2004. *SPF Website*. http://spf.pobox.com/.

APPENDIX

A.   OWL TRUST SCHEMA

```xml
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
    <!ENTITY trust "http://www.konfidi.org/ns/trust/1.4#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY owl  "http://www.w3.org/2002/07/owl#" >
    <!ENTITY foaf "http://xmlns.com/foaf/0.1/" >
    <!ENTITY rel "http://vocab.org/relationship/#" >
  ]>
<rdf:RDF
    xmlns="&trust;" xmlns:owl="&owl;" xmlns:rdfs="&rdfs;" xmlns:rel="&rel;" xmlns:foaf="&foaf;"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
>

<rdf:Description rdf:about="">
    <dc:title xml:lang="en">Trust: A vocabulary for indicating trust relationships</dc:title>
    <dc:date>2006-03-23</dc:date>
    <dc:description xml:lang="en">This is the description</dc:description>
    <dc:contributor>Andrew Schamp</dc:contributor>
    <dc:contributor>Dave Brondsema</dc:contributor>
</rdf:Description>

<owl:Ontology rdf:about="&trust;"
    dc:title="Trust Vocabulary"
    dc:description="The Trust RDF vocabulary, described using W3C RDF Schema and the Web Ontology Language."
    dc:date="$Date: 2005/03/19 11:38:02 $">
    <owl:versionInfo>v1.0</owl:versionInfo>
</owl:Ontology>

<owl:Class rdf:about="&trust;Item" rdfs:label="Item" rdfs:comment="An item of trust">
    <rdfs:isDefinedBy rdf:resource="&trust;" />
    <rdfs:subClassOf rdf:resource="&rdfs;Resource" />
</owl:Class>

<owl:Class rdf:about="&trust;Relationship" rdfs:label="Relationship" rdfs:comment="A relationship between two agents">
    <rdfs:isDefinedBy rdf:resource="&trust;" />
    <rdfs:subClassOf rdf:resource="&rel;Relationship" />
</owl:Class>
<!-- we want to use this for constraints -->
<xsd:element xsd:name="percent" rdf:ID="percent">
    <xsd:simpleType>
        <xsd:restriction xsd:base="xsd:decimal">
            <xsd:totalDigits>4</xsd:totalDigits>
            <xsd:fractionDigits>2</xsd:fractionDigits>
            <xsd:minInclusive> 0.00</xsd:minInclusive>
            <xsd:maxInclusive> 1.00</xsd:maxInclusive>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>

<owl:ObjectProperty rdf:ID="truster" rdfs:label="truster"
    rdfs:comment="The agent doing the trusting.">
    <rdfs:domain rdf:resource="&trust;Relationship" />
    <rdfs:range rdf:resource="&foaf;Agent" />
    <rdfs:isDefinedBy rdf:resource="&trust;" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="trusted" rdfs:label="trusted"
    rdfs:comment="The agent being trusted.">
    <rdfs:domain rdf:resource="&trust;Relationship" />
    <rdfs:range rdf:resource="&foaf;Agent" />
    <rdfs:isDefinedBy rdf:resource="&trust;" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="about" rdfs:label="about"
    rdfs:comment="Relates things to trust items.">
    <rdfs:domain rdf:resource="&trust;Relationship" />
    <rdfs:range rdf:resource="#Item" />
    <rdfs:isDefinedBy rdf:resource="&trust;" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="rating" rdfs:label="rating">
    <rdfs:isDefinedBy rdf:resource="&trust;" />
    <rdfs:domain rdf:resource="#Item" />
    <rdfs:range rdf:resource="&rdfs;Literal" rdf:type="#percent" />
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="topic" rdfs:label="topic">
    <rdfs:isDefinedBy rdf:resource="&trust;" />
    <rdfs:domain rdf:resource="#Item" />
    <rdfs:range rdf:resource="&owl;Thing" />
</owl:ObjectProperty>
</rdf:RDF>
```

B. EXAMPLE TRUST NETWORK

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
<!ENTITY subject "http://www.konfidi.org/example/subject-ns">
]>
<rdf:RDF
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns="http://www.konfidi.org/ns/trust/1.3#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:wot="http://xmlns.com/wot/0.1/">

<foaf:Person rdf:nodeID="alice">
    <foaf:name>Alice</foaf:name>
    <foaf:mbox>demo-alice@brondsema.net</foaf:mbox>
    <wot:hasKey>
        <wot:PubKey>
            <wot:fingerprint>386847DB8862E2262DB3F94EEA6E22F638E76598</wot:fingerprint>
        </wot:PubKey>
    </wot:hasKey>
</foaf:Person>

<foaf:Person rdf:nodeID="bob">
    <foaf:name>Bob</foaf:name>
    <foaf:mbox>demo-bob@brondsema.net</foaf:mbox>
    <wot:hasKey>
        <wot:PubKey>
            <wot:fingerprint>CA1C7BC2FA3AC95EA8AA3E7A1FF947DCC5D954BE</wot:fingerprint>
        </wot:PubKey>
    </wot:hasKey>
</foaf:Person>

<foaf:Person rdf:nodeID="clara">
    <foaf:name>Clara</foaf:name>
    <foaf:mbox>demo-clara@brondsema.net</foaf:mbox>
    <wot:hasKey>
        <wot:PubKey>
            <wot:fingerprint>BB5B0D92A23D31CA559C3D86FF9BD44ADCD8155F</wot:fingerprint>
        </wot:PubKey>
    </wot:hasKey>
</foaf:Person>

<foaf:Person rdf:nodeID="spammer">
    <foaf:mbox>demo-spammer@brondsema.net</foaf:mbox>
    <wot:hasKey>
        <wot:PubKey>
            <wot:fingerprint>ACC267992DDC9AF005D4E24F5013CB50882EC55C</wot:fingerprint>
        </wot:PubKey>
    </wot:hasKey>
</foaf:Person>

<Relationship>
    <truster rdf:nodeID="alice"/>
    <trusted rdf:nodeID="bob"/>
    <about>
            <Item>
                    <topic rdf:resource="&subject;#email"/>
                    <rating>0.90</rating>
            </Item>
    </about>
</Relationship>
<Relationship>
    <truster rdf:nodeID="bob"/>
    <trusted rdf:nodeID="clara"/>
    <about>
            <Item>
                    <topic rdf:resource="&subject;#email"/>
                    <rating>0.70</rating>
            </Item>
    </about>
</Relationship>
<Relationship>
    <truster rdf:nodeID="clara"/>
    <trusted rdf:nodeID="spammer"/>
    <about>
```

```
            <Item>
                    <topic rdf:resource="&subject;#email"/>
                    <rating>0</rating>
            </Item>
    </about>
</Relationship>

</rdf:RDF>
```