Trust Transfer: Encouraging Self-Recommendations without Sybil Attack

Jean-Marc Seigneur[1], Alan Gray[1] and Christian Damsgaard Jensen[2]

[1] Trinity College Dublin, Ireland
Jean-Marc.Seigneur@trustcomp.org, Alan.Gray@cs.tcd.ie
[2] Technical University of Denmark, Lyngby, Denmark
Christian.Jensen@imm.dtu.dk

(This is a revised version of the paper published by Springer-Verlag in the LNCS Volume 3477 / 2005 proceedings of the Third International Conference on Trust Management iTrust 2005. The original publication is available at www.springerlink.com http://www.springerlink.com/openurl.asp?genre=article&id=doi:10.1007/11429760_22)

Abstract. Trading privacy for trust thanks to the linkage of pseudonyms has been proposed to mitigate the inherent conflict between trust and privacy. This necessitates fusionym, that is, the calculation of a unique trust value supposed to reflect the overall trustworthiness brought by the set of linked pseudonyms. In fact, some pieces of evidence may overlap and be overcounted, leading to an incorrect trust value. In this approach, self-recommendations are possible during the privacy/trust trade. However, this means that Sybil attacks, where thousands of virtual identities belonging to the same real-world entity recommend each other, are potentially easier to carry out, as self-recommendations are an integral part of the attack. In this paper, trust transfer is used to achieve safe fusionym and protect against Sybil attacks when pieces of evidence are limited to direct observations and recommendations based on the count of event outcomes. Trust transfer implies that recommendations move some of the trustworthiness of the recommending entity to the trustworthiness of the trustee. It is demonstrated and tailored to email anti-spam settings.

1 Introduction

According to Romano [15], trust is not multiple constructs that vary in meaning across contexts, but a single construct that varies in level across contexts. We adopt this position in this paper as well as her definition of trust:
"Trust is a subjective assessment of another's influence in terms of the extent of one's perceptions about the quality and significance of another's impact over one's outcomes in a given situation, such that one's expectation of, openness to, and inclination toward such influence provide a sense of control over the potential outcomes of the situation" [15].

Throughout this paper, we refer to trust in a given situation as *trust context*. Social research identifies three main types of trust [13]: interpersonal trust, based on past interactions with the trustee; dispositional trust, provided by the trustor's general disposition towards trust, independent of the trustee; and system trust, provided by external means such as insurance or laws. Depending on the situation, a high level of trust in one of these types can be sufficient to make the decision to trust. For example, when there is insurance against a negative outcome, or when the legal system acts as a credible deterrent against undesirable behaviour, it means that the level of system trust is high and the level of risk is negligible - therefore the levels of interpersonal and dispositional trust are less important. It is usually assumed that by knowing the link to the real-world identity, there is insurance against harm that may be done by this entity: in essence, this is security based on authenticated identity. In this case, the level of system trust seems to be high but one may argue that in practice the legal system does not provide a credible deterrent against undesirable behaviour, e.g., it makes no sense to sue someone for a single spam email, as the effort expended to gain redress outweighs the benefit. In this paper, we focus on scenarios where the level of system trust is low. Dispositional trust is reflected in the user setting manually general trust values (for example, for newcomers) and the chosen risk policy (that is, how the user is disposed towards accepting risk). Interpersonal trust is represented as a computed *trust value*. A trust value, that is the digital representation of the trustworthiness or level of trust in the entity under consideration, can be seen as a non-enforceable estimate of the entity's future behaviour in a given context based on past evidence. The basic components of a computational trust engine (depicted in Figure 1) should expose a decision-making component that is invoked when a requested entity has to decide what action should be taken with respect to a request made by another entity, the requesting entity.

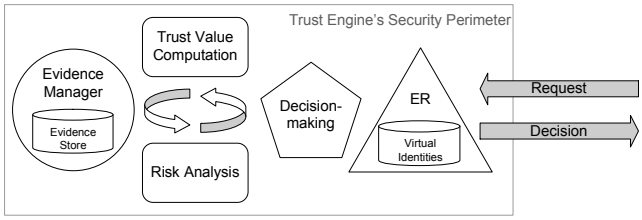

Figure 1. High-level View of a Trust Engine

In order to make the decision to grant the request, two sub-components are needed:

- a trust module that can dynamically assess the trustworthiness of the requesting entity based on pieces of evidence (such as, observations or recommendations);
- a risk module that can dynamically evaluate the risk involved in the interaction.

The chosen action should maintain the appropriate cost/benefit ratio. Depending on dispositional trust and system trust, the weight of the trust value in the final decision may be small.

In the background, another component is in charge of gathering evidence (for example, recommendations or comparisons between expected outcomes of the chosen actions and real outcomes). This evidence is used to update risk and trust information. Thus, trust and risk follow a managed life-cycle. The Entity Recognition (ER [16])

module is in charge of recognising the interacting entities, called virtual identities or pseudonyms.

There is an inherent conflict between trust and privacy because both depend on knowledge about an entity, albeit in opposite ways. Trust is based on knowledge about the other entity: the more evidence about past behaviour is known, the better the prediction of future behaviour should be. However, the more one entity knows about another, the less privacy they have. Although trust allows us to accept risk and engage in actions with a potentially harmful outcome, a computational trust engine must take into account that humans need (or have the right to) privacy. We believe that it is therefore important that any trust engine supports privacy. This is why we have proposed to use pseudonymity as a level of indirection at the identity layer [16], which allows the formation of trust without exposing the real-world identity. It is sufficient to recognise a virtual identity to build trust on past interactions.

However, depending on what benefits can be reaped through trustworthiness, people may be willing to trade part of their privacy for increased trustworthiness: hence, contextual *privacy/trust trade* is possible in our model for privacy/trust trade based on linkage of pieces of evidence [16]. If insufficient evidence is available under the chosen pseudonym, more evidence may be linked to this pseudonym in order to improve trustworthiness and grant the request. In our model, a protocol is given for explicitly disclosing to the requested entity that some evidence can be linked. Some thresholds should be set concerning the acceptable evidence that should be disclosed. An algorithm is used to ensure the *Minimal Linkability principle* [16] – no more evidence than needed to create the link is taken into account.

It may be beneficial to make the link between some pseudonyms explicit (for example, to avoid discounted evidence or reduce the time to reach trustworthiness due to division of evidence between pseudonyms). However, care should be taken when linked evidence on multiple pseudonyms is assessed. The most important requirement is to avoid counting the same evidence twice when it is presented as part of two different pseudonyms or overcounting overlapping evidence. This is especially true when pseudonyms are linked during the privacy trade/process. A *fusionym* is the result of linking two or more pseudonyms together, based on some proof of their linkage, so that they can be regarded as a single pseudonym whose overall trust value is calculated based on the pieces of evidence associated with each composing pseudonym. The solution to calculating the correct trust value for fusionym proposed in this paper works under the following assumptions:

- the trust value is based on direct observations or recommendations of the count of event outcomes from one specific entity; reputation from a number of unidentified entities and credentials generally used in credential-based trust management [18] are not covered by the solution described in this paper;
- the *link* function is only used for linkage at the level of pseudonyms (p): $link(p_1,\ldots,p_n)$ for n linked pseudonyms; the linkage is supposed perfect, unconditionally true and proven by some means beyond the scope of this paper (such as cryptographic signatures, as in [16]);
- a pseudonym can neither be compromised nor spoofed; an attacker can neither take control of a pseudonym nor send spoofed recommendations;
- everyone is free to introduce as many pseudonyms as they wish;
- all messages are assumed to be signed and time stamped.

The next section describes how to achieve safe fusionym based on direct observations and discusses issues introduced by allowing recommendations. Section 3 explains how we use trust transfer to achieve safe fusionym whilst encouraging self-recommendations. The trust transfer approach is tailored to the email and anti-spam sphere in Section 4. Section 5 surveys related work and Section 6 draws conclusions.

2 Fusionym

First we give the formula for the calculation of a fusionym trust value based only on direct observations. We then discuss the issues associated with the use of recommendations in decentralised systems using pseudonyms. Finally, we discuss some of the approaches that others have taken to address the Sybil attack [5].

2.1 Direct Observations

Given the assumptions in the introduction, a trust value is based on the count of event outcomes. Different formats can be used to represent the trust value. A simple scenario is when there is one type of outcome and the trust value is represented as a tuple *(g,b)* counting the number of good (*g*) outcomes and bad (*b*) outcomes. Another format compliant to our assumption is the SECURE trust value [14]. An event outcome count is represented as a *(s,i,c)*-triple, where *s* is the number of events that supports the outcome, *i* is the number of events that have no information or are inconclusive about the outcome and *c* is the number of events that contradict the expected outcome. The trust value consists of a tree of *(s,i,c)*-triples, corresponding to a mathematical event structure. This format takes into account the element of uncertainty via *i*.
With such a trust value format and a perfect linkage, it is sufficient to add all elements of the same type. For example, the fusionym trust value of $link(p_1, \ldots, p_n)$ would be:

$$\sum_{i=1}^{n}(g_i, b_i) \text{ or } \sum_{i=1}^{n}(s_i, i_i, c_i) \text{ in the whole event structure.}$$

2.2 Self-Recommendations

In open environments, security through collaboration allows the computing entities to deal with the high-level of uncertainty by sharing knowledge [14]. Recommendations exchanged between trust engines are the means to share knowledge between the computing entities. Therefore, when the fusionym trust value is calculated, it should be possible to utilise recommendations in addition to direct observations.
However, when recommendations are used and pseudonyms are allowed, *self-recommendations*, that is, recommendations from pseudonyms belonging to the same real-world entity challenge the use of event outcome counts for the calculation of a trust value. In such a system, a real-world entity can use self-recommendations as a low cost mechanism for introducing new pseudonyms in order to protect his/her privacy. If this is the case, then it is difficult to determine how to fairly incorporate this into a trust system based on the count of event outcomes, as the self-recommendation is not based on a history of interactions. If a linkage is performed explicitly after self-recommendations, it may be correct to simply discard the recommendations in the calculation (which means, that the count of recommendations

must be stored separately from direct observation and identified with the pseudonym of the recommender). However, if no linkage is done, then we cannot know if there has been a self-recommendation. Permission to make self-recommendations at will, without cost, paves the way for a Sybil attack, unless explicitly addressed.

2.3 Sybil Attack

A well-known attack in the field of computational trust is Douceur's Sybil attack [5]. Douceur argues that in large scale networks where a centralised identity authority cannot be used to control the creation of virtual identities, a powerful real-world entity may create as many virtual identities as it wishes and in doing so challenge the use of a majority vote and flaw trust metrics. This is possible in trust engine scenarios where the possibility to use many pseudonyms is facilitated by the trust engine. In fact, a sole real-world entity can create many pseudonyms who blindly recommend one of these pseudonyms in order to fool the trust engine. The level of trust in the latter virtual identity increases and eventually passes above a threshold which makes the decision to trust (the semantics of this depend on the application).

One approach proposed to protect against the Sybil attack is the use of mandatory "entry fees" associated with the creation of each pseudonym [1, 6]. This approach raises some issues about its feasibility in a fully decentralised way and the choice of the minimal fee that guarantees protection. Also, "more generally, the optimal fee will often exclude some players yet still be insufficient to deter the wealthiest players from defecting" [6]. An alternative to entry fees may be the use of once in a lifetime pseudonyms (1L [6]), a.k.a. pseudonym commitment, where an elected party per "arena" of application is responsible to certify only 1L to any real-world entity, which possesses a key pair bound to this entity's real-world identity. Blind signatures are used to keep the link between the real-world identity and its chosen pseudonym in the arena unknown to the elected party. However, there are still three unresolved questions about this approach: how the elected party is chosen; what happens if the elected party becomes unreachable; and how much the users would agree to pay for this approach. More importantly, a Sybil attack is possible during the voting phase, so the concept of electing a trusted entity to stop Sybil attacks is fundamentally flawed.

Bouchegger and Le Boudec envisage the use of expensive pseudonyms [6], cryptographically generated unique identifiers and secure hardware modules to counter the Sybil attack. This may overcome the Sybil attack, but at the same time it may exclude poor users as said in [6]. Similarly, Kinateder et al.'s workarounds [10] are two-fold. Firstly, some of the risks of pseudonymity are alleviated via trusted hardware including a trusted certificate agency that would certify the pseudonym without disclosing the real-world identity until legal bodies want to retrieve the link. Secondly, the trust engine should be combined with electronic payment systems, which allow the creation of an originality statement during the payment process which can be included in a recommendation in order to prove that a transaction regarding the recommendation in question really took place. However, relying on real money turns the trust protection into a type of system trust, which is high enough to make the use of interpersonal trust (that is, the trust value) almost superfluous. In the real world, tax authorities are likely to require traceability of money transfers, which would completely break privacy in a money-based system.

A real-world application where Sybil attacks occur is the email system. The success of spammers has proven that it is still cheap enough to create text email addresses, which act as pseudonyms in order to carry out profitable, large-scale spam attacks. It is for this reason that we give an example application of our solution in the email and anti-spam domain in Section 4.

3 Trust Transfer for Recommendations

In addition to considering trust to be a single construct, Romano also notes that trust is "functional, such that trusting behaviours are attempts to attain desirable outcomes by protecting one's interests through actions that either increase or decrease influence in accordance with one' assessment of such influence" [15]. When someone recommends another person, he/she has influence over the potential outcome of interaction between this person and the trustor. The inclination of the trustor with regards to this influence "provides a goal-oriented sense of control to attain desirable outcomes" [15]. So, the trustor expects to be able to increase/decrease the influence of the recommenders according to his/her goals. The goal in this paper is to build a trust engine, which allows recommendations without being vulnerable to the Sybil attack, so the mechanism used to control the recommender's influence must achieve this goal. We conclude that the overall trustworthiness depends on the complete set of trust contexts. This overall trustworthiness must be put in context: it is not sufficient to strictly limit the domain of trustworthiness to the current trust context and the trustee; if recommenders are involved, the decision and the evaluation of the expected outcome should impact their overall trustworthiness according to the influence they had.

La Rochefoucauld [12] highlighted that when one recommends another, they should be aware that the outcome of their recommendation will reflect upon their trustworthiness and reputation since they are partly responsible for this outcome.

With this idea in mind, we argue for the revision of the handling of recommendations in order to achieve safe fusionym and prevent Sybil attacks. In this section, we first examine how others address trust using recommendations. Then we present our model for trust transfer that allows and even encourages self-recommendations from pseudonyms of the same real-world entity. Finally, we present an example of trust transfer in action.

3.1 Adjusted Recommendations based on Recommending Trustworthiness

This is not the first time that the handling of recommendations has been addressed. Since some recommenders are more or less likely to produce good recommendations, the notion of recommending trustworthiness has been added to advanced trust engines. Intuitively, recommendations must only be accepted from senders that the local entity trusts to make judgements similar to those that it would have made itself. Assuming the user has a metric for measuring the accuracy of another sender's recommendations then Abdul-Rahman and Hailes [2], Jøsang [8] and others have suggested models for incorporating that information into the local trust decision. In many cases, the final trust value, which is used locally, may be different to the recommended one. For example, a recommender with trust value of *0.6* on a *[0,1]* scale giving a recommendation of *0.8* provides the discounted trust value:

$$0.6 \times 0.8 = 0.48$$

The overall amount of trust in the system is higher after the recommendation (*0.6+0.8+0.48 > 0.6+0.8*). In a system where there are pseudonyms that can potentially belong to the same real-world entity, a transitive trust process is open to abuse. Even if there is a high discounting factor, the real-world entity can diminish the impact of this discounting factor by sending a huge number of recommendations from his/her army of pseudonyms in a Sybil attack. Additionally, obtaining a measure of recommending trustworthiness is rather difficult, for example, the "semantic distance" [2] between recommendations and local outcomes has to be calculated and that involves the (rather arbitrary) choice of a number of parameters.

It has been noted in the literature that there are issues "with trusting recommenders to recommend arbitrarily deep chains" [2]. They argue that trust at level *n* is independent of trust at level *n+1*. However, this contradicts Romano's view of trust as a single construct that varies across contexts: there is a dependency between trust contexts as they are not independent multiple constructs. Kinateder et al. [10] also take the position that there is a dependence between different trust contexts. For example, a chef known to have both won cooking awards and murdered people may not be a trustworthy chef after all. In the next subsection, we present our view on the dependence between trustworthiness and recommending trustworthiness in the same trust context.

3.2 Trust Transfer

Trust transfer implies that recommendations cause trust on the trustor (*T*) side to be transferred from the recommender (*R*) to the subject (*S*) of the recommendation. A second effect is that the trust on the recommender side for the subject is reduced by the amount of transferred trustworthiness. If it is a self-recommendation, then the second effect is moot, as it does not make sense for a real-world entity to reduce trust in his/her own pseudonyms. So, the overall level of trust in the entire network does not increase via recommendations, in contrast to transitive trust schemes. Even if there are different trust contexts (such as trustworthiness in delivering on time or recommending trustworthiness), each trust context has its impact on the single construct trust value: they cannot be taken separately for the calculation of the single construct trust value. A transfer of trust is carried out if the exchange of communications depicted in Figure 2 is successful. A local entity's *Recommender Search Policy (RSP)* dictates which contacts can be used as potential recommenders. Its *Recommendation Policy (RP)* decides which of its contacts it is willing to recommend to other entities, and how much trust it is willing to transfer to an entity.

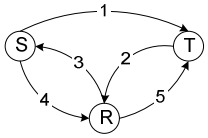

Figure 2. Trust Transfer Process

Trust Transfer (in its simplest form) can be decomposed into 5 steps:

1. The subject requests an action, requiring a total amount of trustworthiness *TA* in the subject, in order for the request to be accepted by the trustor; the actual value

of *TA* is contingent upon the risk acceptable to the user, as well as dispositional trust and the context of the request; so the risk module of the trust engine plays a role in the calculation of *TA*;

2. The trustor queries its contacts, which pass the *RSP*, in order to find recommenders willing to transfer some of their positive event outcomes count to the subject. Recall that trustworthiness is based on event outcomes count in trust transfer;
3. If the contact has directly interacted with the subject and the contact's *RP* allows it to permit the trustor to transfer an amount *(A≤TA)* of the recommender's trustworthiness to the subject, the contact agrees to recommend the subject. It queries the subject whether it agrees to lose *A* of trustworthiness on the recommender side;
4. The subject returns a signed statement, indicating whether it agrees or not;
5. The recommender sends back a signed recommendation to the trustor, indicating the trust value it is prepared to transfer to the subject. This message includes the signed agreement of the subject.

Both the *RSP* and *RP* can be as simple or complex as the application environment demands. In this instance, we limit the policies to simple ones based on trust values. For example, a more complicated *RSP* could be based upon privacy considerations (as is highlighted in the email anti-spam application in Section 4.2). An advanced *RP* could be based upon level of participation in the collaborative process and risk analysis.

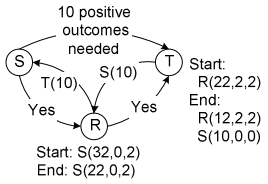

Figure 3. Trust Transfer Process Example

The trust transfer process is illustrated in Figure 3 where the subject requests an action, which requires *10* positive outcomes (recall that the system uses interpersonal trust based on the outcome of past events). The *RSP* of the trustor is to query a contact to propose to transfer trust if the *balance (s-i-c)* is strictly greater than *2TA*. This is because it is sensible to require that the recommender remains more trustworthy than the subject after the recommendation. The contact, having a balance passing the *RSP* (*s-i-c=32-0-2=30),* is asked by the trustor whether he/she wants to recommend *10* good outcomes. The contact's *RP* is to agree to the transfer if the subject has a trust value greater than *TA*. The balance of the subject on the recommender's side is greater than *10 (s-i-c=22-2-2=18)*. The subject is asked by the recommender whether he/she agrees *10* good outcomes to be transferred. Trustor *T* reduces its trust in recommender *R* by *10* and increases its trust in subject *S* by *10*. Finally, the recommender reduces his trust in the subject by *10*.

The recommender could make requests to a number of recommenders until the total amount of trust value is reached (the search requests to find the recommenders are not represented in the figures but the issue is further discussed in Section 4.2). For

instance, in the previous example, two different recommenders could be contacted, with one recommending *3* good outcomes and the other one *7*. A recommender chain in trust transfer is not explicitly known to the trustor. The trustor only needs to know his/her contacts who agree to transfer some of their trustworthiness. This is useful from a privacy point of view since the full chain of recommenders is not disclosed. This is in contrast to other recommender chains such as public keys web of trust [8]. Because we assume for this paper that the entities cannot be compromised, we leave the issue surrounding the independence of recommender chains in order to increase the attack resistance of the trust metric for future work. The reason for searching more than one path is that it decreases the chance of a faulty path (either due to malicious intermediaries or unreliable ones). If the full list of recommenders must be detailed in order to be able to check the independence of recommender chains, the privacy protection is lost. This can be an application-specific design decision.

Thanks to trust transfer, although a real-world identity has many pseudonyms, the Sybil attack cannot happen because the number of direct observations (and hence, total amount of trust) remains the same on the trustor side. Still, local newcomers can be introduced thanks to collaboration. In the previous example, if the subject and the recommender are pseudonyms of the same real-world entity, they remain unlinked. If the proof is given that they can be linked, the fusionym trust value can be calculated as in Section 2.1 with a guarantee of no overcounting of overlapping evidence or self-recommendations.

One may argue that it is unfair for the recommender to lose the same amount of trustworthiness as specified in his/her recommendation, moreover if the outcome is ultimately good. It is envisaged that a more complex sequence of messages can be put in place in order to revise the decrease of trustworthiness after a successful outcome. This is left for future work, because it can lead to vulnerabilities (for example, based on Sybil attacks with careful cost/benefit analysis). The current approach is still limited to scenarios where there are many interactions between the recommenders with the overall trustworthiness in the network (that is, the global number of good outcomes) is large enough that there is no major effect on the trustworthiness of entities when they agree to transfer some of their trust (such as in the email example below). Ultimately, without sacrificing the flexibility and privacy enhancing potential of limitless pseudonym creation, Sybil attacks are guaranteed to be avoided, which is a clear contribution to the field of decentralised, computational trust.

4 Application to Email Anti-spam

In the following subsection, we explain how a trust engine can be used to prioritise emails according to the trustworthiness of their senders. Then, we slightly modify the search for recommenders in order to put more work on the spammer side and take privacy considerations into account.

4.1 Trust value-based Email Prioritisation

There are different techniques to obtain anti-spoofing of email addresses. We assume that each email user uses our Claim Tool Kit [16] proxy, which allows email users to sign their emails based on a key pair. This key pair does not require to be bound to the user's real-world identity. The proxy also runs a trust engine, which is our Java

implementation of the SECURE trust engine [16]. The use of the proxy is transparent to the user, who has just to change the mail server IP address to the IP address of the proxy in his/her preferred email client. The trust value is composed of one triple, which represents the number of emails considered of good quality from the point of view of the email user. The emails in the *Inbox* are prioritised according to the trust value of the sender thanks to a column in the email client graphical user interface ordered by:

$$\frac{s}{s+i+c}$$

In this situation, s and c correspond respectively to the number of good and bad quality emails from the sender. The i element of the triple corresponds to unread email in the *Inbox*. Several unread emails from the same sender are prioritised from the oldest received email to the most recent. Many email addresses can be *pre-trusted* by external means, for example, the email addresses in the *to:*, *cc:* and *bcc:* fields and those appearing in the address book. Pre-trusted email addresses get the value *(1,0,0)*.

If the email is considered too highly prioritised by the user at time of reading, the user can specify in one click that the email prioritisation is wrong and c is increased by *1*; otherwise the sender's s value is increased by one after the email is closed. The trust values are recomputed and emails are reorganised in the folders after each user reading and feedback cycle.

In this example, we assume that compromising email accounts and spoofing are not possible. So, the spammers can only rely on the creation of pseudonyms that will collude to try to get spam emails through. If a spammer sends an email with a disposable email address that will never be reused, the email will end up with the lowest prioritisation, which is *0*. Emails with priority *0* are left in a different folder called *Spam*. Collaboration between email users is used to prioritise emails from legitimate users. Thanks to the user's feedback with regards to the quality of the emails received, the trust value may also be used as the recommending trustworthiness of the sender since senders with higher trust values are likely to prioritise senders of emails of the same good quality.

However, it is possible to engineer advanced attacks on social networks, where the most well-connected users (according to importance metrics such as PageRank, Betweeness Centrality, Degree Distribution Ranker and HITS [9]) are targeted. The success of these attacks hinges on the ability of the attacker to gain information about the topology of the network. This underlines another disadvantage of public online networks of contacts, which goes beyond privacy issues. For example, we mined Google's archive from February 2001 to September 2004 of the *rec.arts.books.hist-fiction* newsgroup. It corresponds to a network of *909* email users. Each contributor to the same email discussion thread is considered a contact of the other contributors to the thread. The threads with only one contributor are discarded. Two contributors appearing in two different threads are not considered to be contacts if they have never contributed to the same unique thread.

Figure 4 gives the overhead of collaboration email when *5* or *25* email users of this network are attacked, either randomly selected or according to an importance metric. The spammer *pleases* them by sending relevant emails from one email address and the remaining users are requested (attacked) by this apparent honest email address, who is considered trustworthy if at least one recommender is found starting from their

direct contacts. If the attacker is considered trustworthy, we say that the user has been *fooled*. Breadth-First-Search (*BFS*) and Random Walk (*RW*) without back-tracking, both limited in number of hops, are compared. Among these Sybil-related attacks, the engineered ones introduce many more collaboration emails, especially when *BFS* is used because the most well-connected nodes have a great number of contacts.

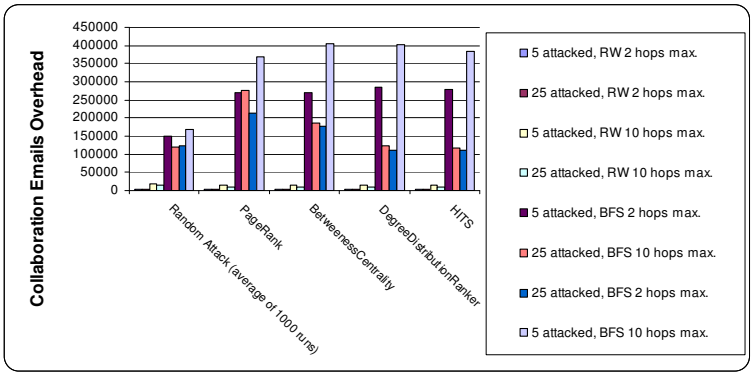

Figure 4. Overhead due to Attacks Caused by No Network Privacy

At first glance, the email infrastructure may be challenged by trust transfer because of the overhead of emails to find potential recommenders. However, in the modified search we present in the next section, the spammer must start the search with his/her own friends. This means that the network load is localised to the spammer's network of honest friends (of whom there are unlikely to be many). As a rule of thumb [1], increasing the cost to the spammer for an attack is desirable and our new trust transfer further increases this cost when engineered attacks are used. In addition to this, recommendations are usually only required for newcomers. Finally, the overhead of collaboration is limited by the *RSP* and more intelligent directed search schemes, for example [7] based on local knowledge about similarity between the sender and the local contacts, do not flood the network.

Concerning the impact of *networked topology engineered* (that we call *NETOPE* in short) attacks, an attack with *5* attacked email users approximately corresponds to *0.55%* of the total email users. An attack with *25* attacked email users approximately corresponds to *2.75%* of the total email users. For *5* attacked email addresses (*0.55%*), the worst case scenario varies from *1.5%* to *91.5%* of fooled email users in random configurations and from *5.8%* to *91.2%* in importance-based configurations. For 25 attacked email addresses (*2.75%*), the worst case scenario varies from *7.4%* to *92.5%* fooled email users in random configurations and from *33.7%* to *91.2%* in importance-based configurations. The greatest difference of impact between the random attack and the engineered attack happens with the following configuration: *5* attacked email users, the *RW* search scheme limited to two hops and PageRank. In the latter configuration, *5.9* times more email users fall in the *NETOPE* attack than in the random attack.

4.2 Proof of Friends (PoF) for Privacy and Sender-borne Search Work

It is not acceptable to leave the legitimate user proxies to carry out the recommender search on behalf of the spammer. So, we revise the trust transfer process in order to put more work on the spammer side. We do not mean it as a strong proof-of-work or bankable postage scheme [1] but it makes more sense to leave the work on the spammer side, where possible. The main idea is to return the list of further potential recommenders to the sender. Instead of the receiver or recommender contacting further recommenders, the sender will use the list to contact each potential recommender (according to the search algorithm chosen).

There is a potential privacy issue in giving the lists of contacts to be processed by the sender. However, since the sender has no other choice to start with his/her own best friends, the lists of potential recommenders can be adjusted according to the trust value of the sender. If the sender is not trustworthy, no list is returned and it cannot find a path to the receiver. In order to ensure non-repudiation, we remind the reader that all requests and responses are signed. Finally, the receiver has to locally verify the signatures (without having to recontact the recommenders). We need another protection mechanism related to privacy disclosure. Each time an email is sent to a new receiver, the email sender sets two local values on a *[0,1]* scale. The first value corresponds to the level of privacy information that the receiver is allowed to see from *0* (none) to *1* (full information). The second value, which is also specified in the sender's email when the receiver must change it, corresponds to the level of privacy required by the contacts of the receiver to be allowed to use the sender as a recommender. A shop may set the second value pretty low since it is in its interest to know as many customers as possible. $Contact_{Privacy}(0.8,0.7)$ means that the contact has a privacy level of *0.8* and allows the recommender to disclose their relationship to subjects with privacy level greater than (or equal to) *0.7*.

Thus, the default trust transfer is changed to the one in Figure 5 (the search requests for the different recommenders are not represented).

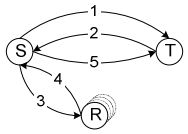

Figure 5. Proof of Friends Trust Transfer

The trust transfer consists of the following steps:

1. The subject requests an action of the trustor;
2. The trustor replies that an amount of trustworthiness, *TA*, is required before it will grant the request;

Until a complete recommender chain is found (or the search's time-to-live expires):

3. The subject starts to query his/her contact email addresses, who pass the *RSP*, to find a recommender chain to the trustor;
4. If the privacy test is passed and the recommender does not know the receiver, it sends back the list of privacy checked contacts to the sender, including a statement signed by the recommender that he/she is willing to recommend the sender as part of a recommender chain, if one can be found;

Once the recommender chain is found, every recommender involved confirms that they have transferred trustworthiness accordingly by signed statement;

5. The subject sends the recommendation to the trustor.

In the example of Figure 6, the sender has only one contact, who does not know the receiver target. However, this contact has one contact who knows the receiver. In this scenario, the *RSP* requires that a potential recommender must have a balance of at least *2TA* on the trustor side. The *RP* is that the subject must have a balance strictly greater than *TA* on the recommender side.

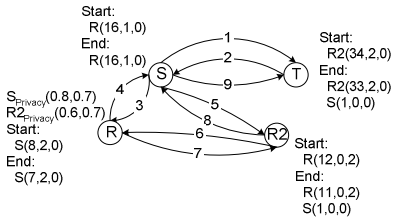

Figure 6. PoF Trust Transfer Example

In our default collaboration scheme, the following emails are exchanged:

1. The sender sends an email to a new receiver;
2. The receiver replies that the proof of having sent one legitimate email is needed to get the email out of the spam folder. In other words, *TA* is *(1,0,0)*;
3. The sender starts contacting his/her list of contacts; in this case, there is only one contact;
4. The sender's only contact is queried; it does not know the receiver, so it starts checking its list of contacts to see if the privacy test is passed; in this case, the sender has a privacy disclosure trust value of *0.8*, which is higher than the threshold specified by the potential recommender *R2*; therefore, the contact is passed back to the sender: more precisely, the contact signs an email stating that it is inclined to recommend *(1,0,0)* for the sender based on a recommender chain including *R2*;
5. The recommender's contact is queried by the sender and is found to be a contact of the receiver; it agrees to recommend *(1,0,0)* for the sender to the receiver as long as it receives a confirmation from *R*. This is because *R* has a balance of *10* on *R2*'s side, which is greater than *TA*. *S* has a balance of *6* on *R*'s side, so the *RP* is passed on all nodes in the recommender chain;
6. An email is sent to *R* in order to confirm that the trustworthiness in the sender has been decreased by *(1,0,0)* on *R*'s side;
7. The trustworthiness in the sender is decreased by one and a confirmation email is sent back to *R2*;
8. *R2* transfers some trustworthiness (one supporting outcome) from *R* to the sender; then, an email confirming that *R2* recommends *(1,0,0)* is sent back to the sender;
9. The recommendation is passed to the receiver, who transfers *(1,0,0)* trustworthiness from *R2*'s trust value to the sender's trust value.

5 Related Work

PoF are pushed recommendations similar to credential chains: "multiple chains can be submitted to establish a higher degree of trust" [18]. However, credential-based trust management misses the possibility of direct observations, which can be used to incrementally form the trust value.

Although we have considered the email system in its current form, others [4] have envisaged deploying structured peer-to-peer layers on top of the email infrastructure and these could be used to facilitate the retrieval of recommenders. However, the change may not be feasible on a worldwide scale as it requires all mail servers to participate. Our proxy solution works for adopters of the proxy, but also for legacy users, although they do not benefit from the anti-spam features. Others have used centralised social networks, such as in TrustMail [7], which does not claim to be a spam filter but prioritises good quality email, as is done in this paper. Our approach is different to TrustMail as it relies on the count of event outcomes, rather than on user defined trust values for prioritisation.

Rahman et al's [2] recommendation protocol is carried out by the trustor and entities involved in the recommendation chain search. We envision that their protocol could be carried out as our PoF protocol, if the trustee is supposed to do most of the work.

Our definition of self-recommendation is different to the one used in [17], where pushed trustworthiness is possible by exhibiting buddy relationships. Roughly, PoF can be considered to be pushed trustworthiness. Furthermore, their buddy relationships may be considered to be wrapped recommendations. So, both their and our approaches provide some kind of pushed recommendations. Finally, successful introduction of local newcomers is possible in both their and our approaches, since pseudonyms with a high PoF can be selected, which is equivalent to a high number of buddy relationships. In contrast to our approach, their approach does not protect against Sybil attacks: to have a high number of buddy relationships does not mean the newcomer is not a spammer because a Sybil attack can be used to fake a great number of buddy relationships between his/her own faked email addresses.

Ziegler and Lausen's trust metric, called Appleseed [19], is possible when it is assumed that all users make their trust values publicly available. Although this can be true in the Semantic Web, due to the publicly crawlable Friend-Of-A-Friend (FOAF) files extended with Golbeck's framework [7] and its trust assertions, this assumption is a threat to privacy since a clear view of the network of acquaintances can be obtained. This is in contrast to our decentralised solution, which permits personalisable levels of privacy. Their trust value ranges from *0* (lack of trust) to *1* (blind trust). Recommending trustworthiness does not explicitly appear, but is reflected in the choice of the "spreading factor" [19], which is recommended to be set to *0.85* and "may also be seen as the ratio between direct trust […] and trust in the ability […] to recommend others as trustworthy peers". Appleseed spreading factor enables trustworthiness gained by an entity to be passed to another entity. If the spreading factor is below *0.5*, most of the granted trustworthiness is kept by the receiving peer and small portion of the trustworthiness is passed to their direct contacts. This is an example of a system where trustworthiness is shared between entities and is in line with the concept of trust transfer. However, this factor is not set by the recommender, but by the computing trustor and is the same for all entities in

the network. In a peer-to-peer way, the recommender should be able to specify the factor itself. An improvement to the scheme may be to use the recommending trustworthiness stated by the receiving peer. In contrast to our solution, their solution creates the problem of the choice of the spreading factor, which, we argue, make no sense to be randomly chosen by the trustor without feedback from the recommender. This is because the recommender is the only entity who has experienced the recommending trustworthiness of his/her contacts.

Ziegler and Lausen [19] also make the distinction between global group trust metrics, which compute a single, global trust value for each entity. This does not allow for the personal bias of any entity and also requires the complete trust network information for all entities. For example, Google's PageRank [3] can be considered here, as pseudonyms and their contacts can be replaced by pages and their hyperlinks. Another type of trust metric, called local [19], takes into account personal bias. Local trust metrics have two sub-types [19]: local group metrics and local scalar metrics. The local group metrics, such as Appleseed, return a subset of the most trustworthy peers from the point of view of the local trustor over a partial view of the trust network, given the amount of trustworthiness desired. Local scalar metrics compute the trust value of a specific entity from the point of view of the local trustor "tracking trust paths from source to target" [19]. Finally, the computation may be centralised or distributed, meaning that the recommendation received is evaluated before being passed to the successor in the recommender chain. Appleseed is a local centralised group metric. Our trust transfer metric is a local decentralised scalar metric.

The OpenPrivacy's "nym manager" [11] allows the user to use different pseudonyms corresponding to public keys. According to the trust context, different pseudonyms can be used. It is possible that a parent pseudonym generates different child pseudonyms. Then, they argue it is possible to prove to an external entity that two children were generated from the same parent anonymously by generating a certificate (or non-anonymously). Of course, they underline that long-lived pseudonyms are preferable in order to be granted interactions requiring a great trust value. In our approach, fusionym is provided to combine the trust values of different pseudonyms once a link has been proven between them.

6 Conclusion

Self-recommendations are the potential means for Sybil attacks in decentralised computational trust systems where virtual identities can be cheaply created and used. Self-recommendations are even a greater issue when the trust engines explicitly support the feature to use multiple virtual identities per user. This enhances the privacy of the system, but it must still be possible to compute accurate trust values.

When trust values are based on direct observations of counts of event outcomes, fusionym provides accurate trust values even if multiple virtual identities are used per user. Still, thanks to trust transfer, safe fusionym is possible even if self-recommendations are used at some stage before the disclosure of the link between the pseudonyms. At the same time, it removes the risk of Sybil attacks.

When evaluated according to the trust metric classification, trust transfer is an instance of a new type of trust metric, local decentralised scalar trust metric.

The evaluation of trust transfer in the email application domain highlights that different search algorithms are possible for trust transfer and some search algorithms are more suitable than others. For example, in email anti-spam settings, it makes sense that the search load is borne by the sender of the email rather than the network, as demonstrated by our proof of friends search for recommenders.
An extended version of this paper, to be published elsewhere, would present that negative recommendations consist of similar trust transfer based on the verification of signed real events recommended to have ended up with a harmful outcome, obviously without the agreement between the trustee and the trustor but followed by a warning of the trustee of this negative recommendation to insist on reciprocity. No incentive has been put in place to force collaboration further since it challenges privacy and some entities are simply not active and not at the centre of the network. Further work including incentives may be interesting in scenarios where privacy is not required.
This work is sponsored by the European Union (IST-2001-32486 SECURE and IST-2001-34910 iTrust) and Enterprise Ireland (grant number CFTD/03/219).

References

1. M. Abadi, A. Birrell, M. Burrows, F. Dabek, and T. Wobber, "Bankable Postage for Network Services", in Proceedings of ASIAN, Springer, 2003.
2. A. Abdul-Rahman and S. Hailes, "Using Recommendations for Managing Trust in Distributed Systems", in Proceedings of the Malaysia International Conference on Communication'97, IEEE, 1997.
3. S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine", in 30(1-7), Computer Networks, 1998.
4. E. Damiani, et al., "P2P-Based Collaborative Spam Detection and Filtering", in Proceedings of the Conference on Peer-to-Peer Computing, 2004.
5. J. R. Douceur, "The Sybil Attack", in Proceedings of the 1st International Workshop on Peer-to-Peer Systems, 2002.
6. E. Friedman and P. Resnick, "The Social Cost of Cheap Pseudonyms", vol. 10(2), pp. 173-199, Journal of Economics and Management Strategy, 2001.
7. J. Golbeck and J. Hendler, "Accuracy of Metrics for Inferring Trust and Reputation in Semantic Web-based Social Networks", 2004.
8. A. Jøsang, "A Subjective Metric of Authentication", in Proceedings of ESORICS, Springer, 1998.
9. JUNG, "JUNG, the Java Universal Network/Graph Framework", http://jung.sourceforge.net/index.html.
10. M. Kinateder and K. Rothermel, "Architecture and Algorithms for a Distributed Reputation System", in Proceedings of the First Conference on Trust Management, LNCS, Springer, 2003.
11. F. Labalme and K. Burton, "Enhancing the Internet with Reputations", 2001, www.openprivacy.org/papers/200103-white.html.
12. La Rochefoucauld, "Réflexions", 1731.
13. D. H. McKnight and N. L. Chervany, "What is trust? A Conceptual Analysis and an Interdisciplinary Model", in Proceedings of AMCIS, 2000.
14. N. Mogens, M. Carbone, and K. Krukow, "An Operational Model of Trust", SECURE Deliverable 1.2, 2004, http://secure.dsg.cs.tcd.ie.

15. D. M. Romano, "The Nature of Trust: Conceptual and Operational Clarification", in PhD Thesis, Louisiana State University, 2003.
16. J.-M. Seigneur and C. D. Jensen, "Trading Privacy for Trust", in Proceedings of the Conference on Trust Management, Springer-Verlag, 2004.
17. F. Stefan and O. Philipp, "The Buddy System - A distributed reputation system based on social structure", Technical Report 2004-1, Universitat Karlsruhe.
18. W. H. Winsborough, K. E. Seamons, and V. E. Jones, "Automated Trust Negotiation", DARPA Information Survivability Conference, 2000.
19. C.-N. Ziegler and G. Lausen, "Spreading Activation Models for Trust Propagation", in Proceedings of the International Conference on e-Technology, e-Commerce, and e-Service, IEEE, 2004.