# A Non-Transitive Trust Model for Key Distribution

Sarvjeet Herald, Stephen Clarke and Bruce Christianson
School of Computer Science, University of Hertfordshire
College Lane, Hatfield, AL10 9AB, UK
{s.herald, s.w.1.clarke, b.christianson}@herts.ac.uk

*Abstract:* Key distribution mechanisms such as PKI or PGP implicitly assume trust to be transitive. This can be a problematic assumption. The user relies indirectly (often implicitly) on the remote entities to satisfy its trust requirements. In fact, over the years trust has been a much debated topic in the electronic world. In our view, trust is most usefully modeled as non-transitive and subjective to the user. This paper explores a novel way to address the well known asymmetric key distribution problem in the electronic world by mitigating the subjective risk of the user. We extend the conventional PKI and PGP models by deploying a recently introduced concept called trust*. Trust* is a way of building on existing trust relationships using an electronic equivalent of real-world guarantees so as to avoid the need for transitive trust. This application of trust* provides a flexible way to bridge the gap between the two unknown entities through the use of localized guarantees. Our model allows trust* to replace the need for transitive trust in PKI or PGP and thus reduce the perceived risk of the user in key distribution.

## 1 Introduction

Trust [11], [12] is a much debated topic in the electronic world. The electronic world is very different from the real world as the trusting party is usually unqualified to evaluate risks in the electronic context.

Due to an enormous growth of the Internet, it is highly likely that users do not have a prior relationship but still want to communicate. In such scenarios, users with no personal relationship can securely exchange messages using public-key cryptography. A unique public-private key pair can ensure, for example, that their messages are confidential: if the message is encrypted with the public key of the recipient and sent to the recipient over an open communication channel, only the owner with the corresponding private key will be able to decrypt it. However, an interesting problem faced by the growing number of electronic users is to find the correct public key of the recipient. This problem is often referred to as the key distribution problem. Existing solutions such as Public Key Infrastructure (PKI) [2] or Pretty Good Privacy (PGP) [22] involve a third party (assumed to be trusted) like a public-key authority, public-key directory, key signing or reputation system for key distribution. The problem with existing solutions is they implicitly assume trust to be transitive. This can be a problematic assumption. The user relies indirectly (often implicitly) on the remote entities to satisfy its trust requirements.

In a recent work, Clarke, et al. [6] discuss how analogies to real world guarantees via intermediate entities can be used to replace the need for transitive trust. This paper will discuss how the need for transitive trust can be eliminated for key distribution. The notion of trust used in this paper is first-hand subjective trust based on the user's already established personal relationships. The application of Clarke's concept of trust* can be used to bridge the gap between two unknown entities, thus reducing the perceived risk of an individual.

## 2 Trust

Over time researchers have come up with definitions of trust as transitive, derived, bi-directional, or relied upon [11], [13], [14], [21]. However, it is often argued that trust is not always transitive [5]. On the Internet, within and across domains, trust can be broadly classified into three trust models: Direct Trust, Hierarchical Trust and Web of Trust [4].

### Direct Trust

All cryptosystems use direct trust in some way. This simple trust model is based on a direct relationship one individual has with another based on an existing relationship. In the real world, direct trust is often one-way, for example, Alice trusts Bob, but Bob does not trust Alice. Similarly on the Internet, trust should be generally assumed one-way. But in many situations direct trust is assumed to be two-way.

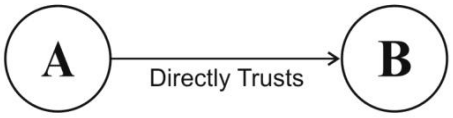


Figure 1: Direct trust model.

### Hierarchical Trust

Hierarchical trust allows two principals in different administrative domains to communicate if a chain of trust is established between the two. Usually each domain has its own Certification Authority (CA). This trust model represents a

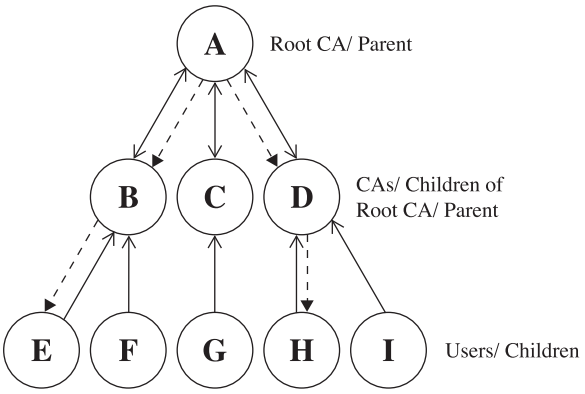


Figure 2: Hierarchical trust model.

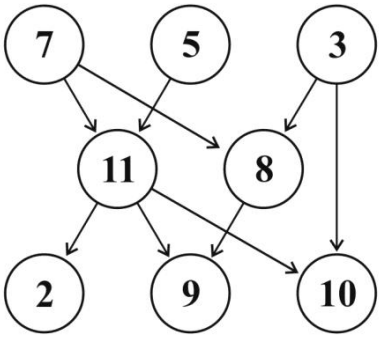


Figure 3: Web of Trust

parent-child relationship where a single parent can have multiple children who trust their parent. Hierarchical trust is commonly used in PKI for cross verification of Public Key Certificates (PKC). It helps users with no prior relationship to build (transitive) trust relationships. Figure 2 shows a typical parent-child relationship in a hierarchical trust model. It shows a PKI with a root CA, other CAs and the end users.

Parents have knowledge about a child's key. Every child sends its key to the parent who registers the key in its directory (after verifying the child according to the parent's policies and procedures). Figure 2 shows the complexity of the trust assumptions implicit in the hierarchical trust model. The dotted arrow shows that a parent has the knowledge of the public-key of the child. Solid arrows show direct trust. Having knowledge about someone is very different from trusting someone.

Suppose that E wants to communicate with H, then A is the lowest common entity between the two. E trusts B to give him the correct key of H, but B does not know the key of H. B trusts A to give B the key of H, but even if we assume that trust *is* transitive (whence E trusts A to give B the key of H) this is still not enough: A does not know the key of H either. D does know the key of H, but now as well as transitivity we also need to assume symmetry of the trust between CAs: A must trust D to give A (and hence B) the correct key of H. These assumptions on trust (which imply that every end user must trust every CA) are often not justified, and usually are not even stated explicitly, for the PKI.

### Web of Trust

The concept of a Web of Trust was first put forth by PGP [18], [22] creator Phil Zimmermann in 1992 where the responsibility of validation of public keys is delegated to people you trust. The Web of Trust is a general directed graph and does not have a centrally trusted controller (such as a CA) but instead depends on the trust of other users. The public key is signed by a trusted person who acts as an introducer. The solid arrow in Figure 3 represents direct trust. For example, 11 or 8 may introduce 9 to 7 due to direct trust (7 directly trusts 11 or 8 and 11 or 8 has knowledge of the key of 9). The final trust decisions are left to the user based on his trustee's assertion that the public key in circulation is correct.

Phil Zimmermann did not assume trust to be transitive in web of trust. The trust is limited to the introducer. But, the users of the web of trust in PGP made an assumption of trust transitivity and extended web of trust to allow chains of introducers.

## 3 Key Distribution Mechanisms

To develop trust among users with no prior relationship, various key management mechanisms are used in practice. Dent and Mitchell [9] discuss standards for key management, a key's life cycle and its usage. Often a trusted third party, PKI, PGP or reputation system is used to increase trust on the Internet and solve the key distribution problem. However, relationships between unknown parties are based on second hand (i.e. indirect) trust. The truster does not know the trustee personally. Trust (often implicitly) is thus assumed to be transitive. A trustee has to rely on (an assumed) trusted third party to provide assurance on the public-key of the recipient. This is due to lack of alternatives for the user and can be a problematic assumption.

### Public Key Infrastructure (PKI)

The primary principle of PKI is to establish trust hierarchies to enable secure, convenient, and efficient acquisition of public keys. Stallings [19] states: "RFC 2822 (Internet Security Glossary) defines public-key infrastructure (PKI) as the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography." Typically multiple CAs are involved in a PKI, responsible for generation, management, storage, deployment, and revocation of public key certificates and require cross verification. Vacca [20] states: "When CAs negotiate cross-certification services, they will examine each others CPSs[1]. The liability of the certificate issuers and the end entities together helps in the degree of the trust."

When designing the PKI, it was thought that these hierarchies would provide a greater degree of trust. Ellison and

[1]Certification Practice Statement

Schneier [10] discusses the risks of PKI and the imprecise use of the word "trust". The CA's are actually the organisations used to establish trust. There are few popular CA's on the Internet, but in fact, it is very easy for anyone with skills to behave as a CA and issue certificates. This makes it difficult for a user with no personal relationship with a particular CA, to analyze the threats regarding the ownership and authenticity of a public key.

Jøsang [13] recommended the introduction of relative trust in the key to owner binding, and recommendations based on first-hand evidence, as a way to transitively trust which mitigates the risk involved in PKI. However, the element of risk in existing public-key based trust relationships arises from the fact that trust is assumed to be transitive. The risk increases with the increase in the complexity of the hierarchies. The relying party relies indirectly (often implicitly) on the remote CA (the assumed-to-be-trusted third party) to verify that the principal carrying the key is the owner of the key.

### Pretty Good Privacy (PGP)

PGP [1], [18], [22] is a tool intended to provide Internet users with cryptographic privacy and authentication. It removes the need for hierarchies of certification authorities (CAs). The approach is a decentralised web of trust where individuals sign the key certificates of others. Key legitimacy of an unknown entity is provided to the sender by a trusted entity (who also acts as an introducer to the sender). The introducer recommends that the key in the certificate belongs to the person stated. A web of trust is a more generic solution than the hierarchical approach, but potentially more difficult to manage [4]. The individuals who sign the key may still be unknown to the truster. Moreover, in PGP trusting a key is not the same as trusting a key's owner [16].

In PGP, multiple chains of individuals are formed between end entities while recommending a key. This often results in a problem, "hidden dependencies in PGP trust values" [13]. The web of trust in PGP causes the transfer of recommendations between users based on second hand evidence, which may lead to violation of trust requirements of the verifier.

### Reputation Systems

A reputation system [3], [17] collects, distributes and aggregates feedback about participants past behavior to predict future behavior to each transaction. Reputation systems seek to establish trust between strangers who hold no relationship but want to communicate. For example, PGP has its own type of reputation system called "web of trust" to solve the problem of uniquely identifying public-key certificates and who they belong to. A reputation system tries to increase trustworthiness between strangers. Users can append ratings to the key, which would enable strangers to trust the public-key certificate. Usually a user has no personal relationship with the entity recommending. But even when they do, users must still trust the public-key certificate based on second hand evidence. This has the effect of requiring transitivity of trust. Although the final trust decisions are left on the user, the lack of alternatives forces the user to trust the recommendations.

## 4 Trust*

The trust* model [6], [7], [8] uses guarantees to extend local trust between end-points to a new relationship called trust*. This is analogous to a real world protocol, where a guarantor can be used to replace the need for transitive trust. (For example, the buyer trusts the guarantor, the guarantor trusts the vendor. If the vendor defaults, the guarantor will compensate the buyer, who thus does not need to trust the vendor. All trust remains local.) Trust* extends local trust to a new relationship which can hold between principals that are unknown to and do not trust each other. Guarantees are provided by principals within pre-existing (local) trust relationships. Each principal directly trusts the principal providing them with a guarantee, but there is no end-to-end trust. The pre-established trust relationship mitigates the subjective risk of a user.

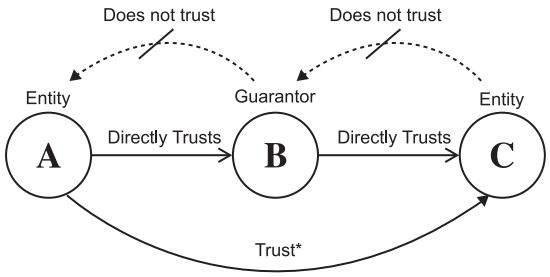


Figure 4: Trust* relationship.

Trust* is based on a unidirectional chain of locally trusting principals and can consist of a arbitrary number of hops to communicate with the unknown entity (Figure 4). Each trust relationship is localised and one-way, and so trust* can be extended to multiple guarantors between the end entities. Risk is redistributed to the guarantor B who will take the responsibility because he trusts entity C based on his long term relationship. The guarantor B provides a guarantee to A that C will act appropriately. If C does not, then B will pay A a forfeit. Guarantees are localised and one-way, so reverse trust is not required, Entity C need not trust the guarantor B.

A payment model is used for the provision of guarantees in the trust* protocol. These include commission payments as an incentive for giving a guarantee and forfeit payments as a deterrent for defaulting a guarantee. Forfeits may also compensate the affected party. These payments are paid locally and can be micro-payments, CPU time, database access, bandwidth or something else of local value.

The concept of extending trust using localised guarantees enables the trust*er to act *as if* he trusts the trust*ee directly. As actual trust relationships are localised, trust management and payment mechanisms can be heterogeneous and end entities participating in the trust* relationship can even remain anonymous to each other. An important advantage of trust* is that no new trust relationships need to be built for trust* to work.

## 5 Trust* Key Distribution

On the Internet where users have no existing relationship, the introduction of trust* to the key distribution mechanisms discussed earlier mitigates the subjective risk of the user. The trust*er (user) develops a trust* path to the trust*ee via a guarantor who he directly trusts based on his existing trust relationships. The user contacts his trusted source for a guarantee of the end-point. The guarantor acts as a mediator to develop a trust* path between the end entities. This eliminates the need for transitive trust in the existing key distribution mechanisms. The use of forfeits as deterrent provides assurance to the trust*er. The commission is an incentive to the guarantor to provide the guarantee.

### PKI with Trust*

The user contacts a particular CA who issued the public-key certificate to the entity he wants to communicate with, using a guarantor. Now, instead of the complex trust relationships in hierarchical trust model (figure 2), the hierarchies of trust are reduced to single hops by trusting only the principal providing the user with a guarantee. The introduction of trust* to PKI avoids unnecessary trust assumptions and eliminates the transitive trust. The user E need only find a guarantor which he trusts directly to guarantee the public-key certificate from the CA D. The user E and the guarantor agree on commissions and forfeit rates before he receives the public-key certificate from D, through the trust* relationship. The guarantee mitigates the subjective risk of the user and provides assurances on public-key certificate. The user need not trust the CA D, he trusts only the guarantor. The direct trust of the user on guarantor can be revoked or updated anytime if the trust ceases or changes.

### PGP with Trust*

Similar to the early web of trust, trust now is limited to one hop. The trust complexity of the chain of introducers is reduced to the (pre-existing) relationship with a guarantor who guarantees the end node. This eliminates the need for transitive trust and mitigates the subjective risk of the user. There can even exist end-to-end anonymity as the user need not have knowledge about the end node because he is assured with the guarantee. In order to enter into a trust* relationship with the end node, the user only needs to find a guarantor who has a trust* path to the end node and agree on commission and forfeit rates. The user does not trust the end node, but relies on the guarantee. As trust is now localised, the use of trust* removes the hidden dependencies in trust values discussed earlier. As with PKI, the direct trust the user has with guarantor can be revoked or updated anytime if the trust ceases or changes.

### Routing with Trust*

Any established network routing protocol can be used to find an appropriate chain of guarantors in a trust* relationship. This requires no change to the current infrastructure. Routing decisions are often based on two popular algorithms: Dijikstra Algorithm and Bellman-Ford Algorithm; involving least cost or distance vector [8]. Routing algorithms such as RIP v1, RIP v2, IGRP, EIGRP, OSPF, BGP, PBR [15] can be used to establish a trust* path between the end-entities. However, we propose a specific algorithm, called Web of Trust*, that uses trust* for key distribution.

## 6 Web of Trust*

Web of Trust* does not have a centrally trusted controller, and does not rely on a third party or an introducer to develop new trust relationships on the Internet. Unlike web of trust in PGP, assurances are provided by the guarantor who the user directly trusts. Trust* is extended between the end-entities using a guarantor (or chain of guarantors). Trust does not go beyond one hop and is always local. End-entities need not trust each other directly, but instead use trust*. Commission and forfeit are used as a cost metric (similar to metric in other routing algorithms) that help in deciding the route. We propose a particular algorithm as a part of our Web of Trust* which mitigates the subjective risk of the user in key distribution. This algorithm is motivated from flooding and adaptive algorithms, and is used to build trust* relationships for key distribution.

### The Algorithm

Shortest path is often not the most secure path. Unlike Open Shortest Path First (OSPF), instead of finding the shortest path between the end entities, the route is found using the local directly trusted nodes. Trust* can then extend this to any number of hops to reach the end entity. Each node first checks its trust* table (Figure 5) to find any already established trust* relationship with the trust*ee. A trust* table maintains the state of the already established trust* relationships and the guarantor. If an appropriate trust* relationship is found at any node $(n)$ then there takes place a complete transfer of this part of the trust* table to node $(n-1)$, if the node $(n)$ is willing to provide a guarantee. This is similar to distance vector protocols such as RIP, IGRP where the entire table is broadcast to the neighbor. The difference is due to localization, each node receives first hand information instead of second hand and the trust* table is unicast not broadcast. Each node maintains trust* state locally.

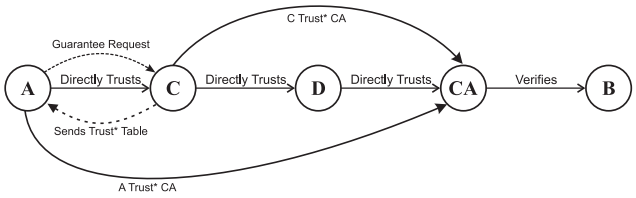


Figure 5: Web of Trust*

The optimal trust* route chosen depends on the following:

**Case 1: No suitable guarantor to provide guarantee.** If there are no already established trust* relationship in the trust* table, the client sends a guarantee request message to all its directly trusted nodes. Instead of flooding the network, guarantee request are sent only from node $(n)$ to the node

$(n+1)$ where node $(n+1)$ are the nodes listed in node $(n)$'s direct trust table. Each node repeats the algorithm.

**Case 2: The guarantor is not willing to provide a fresh guarantee.**
Even if there is an already established trust* relationship, the guarantor might reject the new request. This can be for various reasons: he no longer trusts the node for which he was guaranteeing, his guarantee policy does not allow him to provide another guarantee to the client, etc. The client then behaves *as if* there is no already established trust* relationship, removes the trust* relationship from the trust* table and starts looking for new guarantor (Case 1).

**Case 3: The guarantor is willing to provide a fresh guarantee.**
A guarantor might be willing to provide a fresh guarantee if there is an already established trust* relationship, after agreeing on commission and forfeit rates. Then a complete transfer of the trust* table in the reverse trust* path takes place. The node $(n-1)$ on receiving the trust* table from node $(n)$, first updates its trust* table and then sends its updated trust* table to the node $(n-2)$. This process is repeated until the client node receives the updated trust* table and updates its own trust* table.

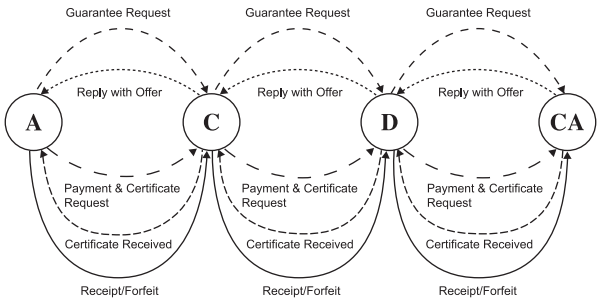


Figure 6: Message passing in Web of Trust*

Figure 6 shows the message passing in Web of Trust* to get the public-key certificate from CA.

1. The client A requests a guarantee from its directly trusted node C (possible guarantor) depending upon *if* there is an already established trust* relationship or not, in the trust* table. Checking the trust* table first is necessary.
2. If the directly trusted node C is available to provide a guarantee, it replies with an offer i.e. commission and forfeits rates (this may involve negotiation until the client accepts the offer).
3. If client A accepts the offer, he pays the up-front commission payment to the guarantor C and requests the public-key certificate.
4. The client A receives the public-key certificate from the guarantor C.
5. The client A sends either a delivery receipt to the guarantor C or claims forfeit.

Similar to link state protocols, message passing is done only when a guarantee is required. It is unnecessary to send advertisements on changes in the state, when no guarantee is required. If the guarantor is willing to give a fresh guarantee, it then sends its new guarantee conditions i.e. commission and forfeit rates. Commission and forfeit are used as a cost metric similar to the metric used in other routing algorithms that help in deciding the route. If the new guarantee conditions are acceptable to the client, the client pays up-front commission payment and requests the public-key certificate. A guarantor, on receiving the public-key certificate request from the client, requests the public-key certificate from the end-entity (via another guarantor, if involved). The end-entity passes the public-key certificate back to the requester, which eventually reaches the client. A copy of the public-key certificate is maintained at each node until the client sends either a successful delivery receipt or claims a forfeit.

### Advantages of Web of Trust*

1. The node knows many other possible trust* routes through the passing of trust* tables.
2. The node is in a better position to provide guarantees to the nodes that directly trust him, due to the increase in knowledge of more routes.
3. The majority of requests only have to go one hop to find a suitable trust* route, once the trust* routes are established.
4. When a node in a route is down, the guarantor is responsible for attempting to find the new route for his client. In the case that the guarantor is not responding, the client can try another guarantor.

There is a small load in the network due to the complete transfer of the trust* table. But in subsequent searches for a guarantor, a client has prior knowledge and this prevents network congestion.

## 7 Implementation

A prototype of this model was implemented to integrate trust* with PKI using web of trust*. A client on receiving public-key certificate from the guarantor, verifies it to ensure the certificate is valid against the basic minimum requirements. Alternatively, the guarantor can first check the certificate before delivering it to client, but this unnecessarily increases the work load for each of the guarantors involved in the trust* path. The following checks are performed on a certificate:

1. The certificate received is of the correct owner as requested.
2. The certificate (or any part of the issuing chain) is not expired.
3. The certificate is not listed in a certificate revocation list.
4. The signature attached to the certificate is valid i.e. the hash value matches.
5. The certificate does not violate the key usage or the basic constraints set by the CA.

6. The purpose of the certificate issued is same as needed.
7. The names of an issuer and a subject must form a chain, i.e. except the first and last certificates, a subject of a higher certificate must be an issuer of a next certificate.
8. The CA issuing the certificate is in current certificate store and enabled. This validation is repeated until the root certificate is validated.
9. The signing key matches with the public key in the certificate.
10. The decrypted contents of the MIC[2]-Info field matches the MIC computed locally.

If the certificate fails the above test, the client claims the forfeit from the guarantor. Depending on the importance of the communication for which the guarantee was sought, the client may cease his direct trust or look for another guarantor from his directly trusted sources. The client provides the reason for the forfeit claim such as, "the certificate received is expired", etc. The guarantor, who fears false forfeit claim requests, repeats the above test, matches the reason of claim and checks the certificate himself. He uses the locally stored copy of the certificate to ensure that the claim is genuine. The guarantor always carries the risk of losing his future earning if he does not pay a genuine forfeit claim. A client might cease to trust him. The guarantor loses his future incentives: profit earned by providing correct guarantees.

If the guarantor finds the certificate to be correct, he requests a fresh certificate from a different trust* route, checks the certificate and matches the two certificates to find the genuineness of his guarantee and also of his guarantors, if any. The guarantor can then make better decisions: pay the forfeit, negotiate with the client or simply deny the wrong forfeit claim and take the consequences.

If the guarantor repeatedly does not pay the client the agreed forfeit, this eventually results in cease of trust from the client. According to the guarantor this might be a better option for him as he senses a greater threat in giving future guarantees. On the other hand, if the client still trusts the guarantor, and requests a guarantee again, the guarantor might not offer a guarantee (after analysing his risks) or may demand higher commission.

### Negotiation

Analogous to the real world, a guarantor may try to convince the client to withdraw a forfeit claim and not to lose future commission. If the policy allows, a guarantor might decide to enter in negotiation with the client. The guarantor sends a negotiation message to the client stating his reason for the correctness of the certificate. If the client is still not convinced, and demands the forfeit a second time, the guarantor may consider his request to oblige him. This is analogous to a customer not convinced with the quality of the goods purchased and visits the store to claim a refund. The store looks into its company's policy and tries to convince the customer with valid reasons. They may know the customer is wrong but to oblige him, may offer him a replacement or an exchange on other goods. A fuzzy customer might still want his money back. So, the store (who wants to oblige and retain the customer for future profit) will likely refund the money. A persistent complainer will eventually not get a refund.

The guarantor may offer the client a free guarantee from a different trust* route to ensure him the correctness of the certificate. If the client accepts the offer, the guarantor sends him another trust* route. The client and the guarantor record this act of obliging. A fuzzy client still not convinced demands the forfeit, then depending on the risks involved in giving back the forfeit: potential profit and customer retention; a guarantor might return the forfeit but records it. However, the maximum permissible refunds before ceasing the relationship depend on the individual guarantor because he does not trust the client. The client on the other hand might eventually consider this a breach of trust and cease his trust in the guarantor anyway.

## 8 Discussion

An advantage of using trust* is heterogeneity: because all trust is local, each user can apply his own authentication mechanisms. Any already established authentication mechanisms can be employed to authenticate the two users. Generally, a user signs the message with his private key, and sends it to the guarantor after encrypting the message with the public-key of the other end.

The need for end-to-end anonymity depends on the key distribution mechanism in use. By using trust*, end-entities may remain anonymous. For example, the client's pre-knowledge of the end point (i.e. CA who issued the public-key certificate) need not require end-to-end anonymity for key distribution in PKI. The motive is to find a trusted source who can guarantee the public-key certificate from a particular CA. While in PGP, the uncertain knowledge of the end-point motivates end-to-end anonymity. A client is far-less concerned with the end-point as long as he is getting assurance from his trusted source.

Analogous to the real world, a guarantor is willing to provide a guarantee for monetary benefits. This model ensures that the electronic guarantor gains incentives for his guarantee. Before the trust* relationship can be established, the client and its directly trusted node (the guarantor), agree on commission and forfeit type and rates.

Commission is paid up-front. This incentive can also be considered as an act of obliging the guarantor in the hope to get more successful guarantees in future. If the client is not satisfied with the guarantee he may demand the agreed forfeit otherwise he will notify the guarantor that he is happy with the guarantee. The commissions and forfeits are the building blocks of trust* where failure of payment may result in the cease of trust or change in relationship. The forfeit acts as a deterrent to discourage false guarantees and prevent the delivery of false certificates to the client.

Short-term problems may arise when a guarantor is receiving a commission from his client and also from the entity he is guaranteeing. If the guarantor is getting more commission from the entity he is guaranteeing than the forfeit he is paying to the client, the guarantor's changed incentive might result in

[2]Message Integrity Check

giving a wrong guarantee to the truster. But if the client has to claim the forfeit regularly from this guarantor, the client will eventually decide to cease his local trust. Alternatively, a guarantor may decide to suddenly increase the client commission to avoid frequent forfeit claims from the same client, rather than simply cutting off the guarantee requests.

Trust* is one-way: the client trusts the guarantor, but the guarantor need not trust the client. A guarantor might be in doubt, if the client is regularly falsely claiming the forfeit. Similar to the trust* table, a reverse trust* table helps in deciding on future guarantees by the guarantor. A guarantor records the service he provided to the truster which includes commissions and forfeits. A guarantee policy (if needed) can be associated with this reverse trust* table. If the guarantor has already provided a guarantee to the client, then a fresh guarantee should only be provided after checking the reverse trust* table.

In the situations where the trustee suspects the truster of false forfeits claims, he can choose an additional route from a different direct trust relationship. This can provide assurances to the guarantor or the end-entity who otherwise do not trust the truster.

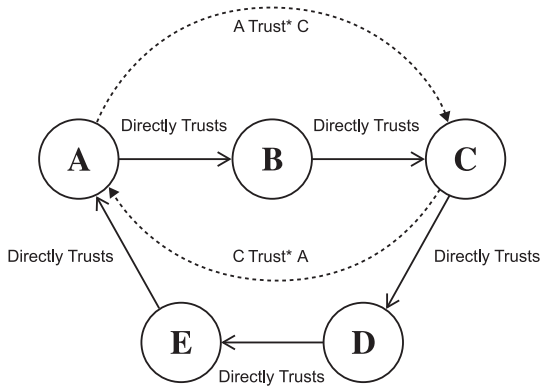


Figure 7: Cycles of Trust*

"Cycles of trust*" [8] can be used to further decrease the subjective risk. The cycles of trust* extend the Web of Trust* in the opposite direction. If a trust* path can be found from A to C, then it is usually possible to find a trust* path from C to A, via different sequence of nodes. This protects against spoof forfeit claims. Figure 7 is an example where B directly trusts C but C does not trust B or A. In such situations, the Web of Trust* can be extended to C's pool of directly trusted nodes to find a guarantor for a reverse guarantee that false forfeit claims will not be made. This will assure authenticity of A to C, decrease the subjective risk of C on false claims, and prevent false/spoofing guarantee requests.

Other issues such as congestion can be considered in the future. For example, when particular nodes increase their prices, this may cause "nearby" nodes to overload, particularly when long trust* paths involving many nodes are affected.

There is more to be done, to make this approach commercially viable, efficient and scalable, depending on the reader of this paper.

# 9 Conclusion

Trust is a much debated topic on the Internet and is defined in different ways. We believe, trust is not always transitive. Existing key distribution mechanisms implicitly assume trust to be transitive. The user is forced to rely on third parties for the correct public-key. This is due to the lack of alternatives currently available to the user. Trust* is an alternative approach which eliminates the need for transitive trust. The motive behind this research was to explore a new way of building on the trust in conventional key distribution mechanisms and which avoids the need for transitivity of trust.

Trust* uses guarantees to extend the effects of local trust to end-points. All trust is local; end to end authentication is not required. Indeed the end-entities may remain anonymous. The Web of Trust* introduced in this paper establishes a trust* path between the end-entities. Then the trust*er can act *as if* he trusts the trust*ee directly. This algorithm maintains the state of the trust* path locally at each node. There is a small load in the network due to the complete transfer of the trust* table. But in subsequent searches for a guarantor, the client has prior knowledge and this reduces overall network load.

The commissions and forfeits in trust* are used as incentive and deterrent to encourage delivery of the correct public-key. Forfeits can also ensure that the affected party is compensated. The commission is paid up-front and the client on receiving the certificate either sends a message to guarantor that he is happy or claims a forfeit.

Trust* requires no change in infrastructure and has the benefit that it builds on already established trust relationships. No new trust relationships need to be built. Trust management remains local, thus unlike other methods that address the key distribution problem, trust* is heterogeneous in many contexts.

This research shows that the combination of trust* with either conventional PKI or PGP is feasible. The need for transitive trust is eliminated. The trust required is now localised. Thus, the subjective risk perceived by the user is decreased.

# References

[1] Alfarez Abdul-Rahman. The pgp trust model. Technical Report. University College London, 1996.

[2] Erik Andersen. The x.500 directory standard: A key component of identity management. *Telektronikk*, 1:160–164, 2008.

[3] G. Bella, G. Costantino, and S. Riccobene. Evaluating the device reputation through full observation in manets. *Journal of Information Assurance and Security*, 4(5):458–465, 2009.

[4] Germano Caronni. Walking the web of trust. In *WETICE '00: Proceedings of the 9th IEEE International Workshops on Enabling Technologies*, pages 153–158, USA, 2000. IEEE Computer Society.

[5] Bruce Christianson and William S. Harbison. Why isn't trust transitive? In *Proceedings of the International*

*Workshop on Security Protocols*, pages 171–176, London, UK, 1997. Springer-Verlag.

[6] Stephen Clarke, Bruce Christianson, and Hannan Xiao. Trust*: Using Local Guarantees to Extend the Reach of Trust. In *Proceedings of the Seventeenth International Workshop on Security Protocols*, April 2009. To appear.

[7] Stephen Clarke, Bruce Christianson, and Hannan Xiao. Extending Trust in Peer-to-Peer Networks. In *Advances in Databases and Information Systems, LNCS 5968*, 2010.

[8] Stephen W Clarke. *Trust*: Extending the Reach of Trust in Distributed Systems*. PhD thesis, University of Hertfordshire, 2009.

[9] Alexander W Dent and Chris J Mitchell. *User's Guide to Cryptography and Standards*. Artech House, 2005.

[10] Carl Ellison and Bruce Schneier. Ten risks of pki: What you're not being told about public key infrastructure. *Computer Security Journal*, XVI(1):1–7, 2000.

[11] Tyrone Grandison and Morris Sloman. A Survey of Trust in Internet Applications. *IEEE Communications Surveys and Tutorials*, 3(4):2–16, 2000.

[12] Jingwei Huang and David Nicol. A calculus of trust and its application to pki and identity management. In *IDtrust '09: Proceedings of the 8th Symposium on Identity and Trust on the Internet*, pages 23–37, USA, 2009. ACM.

[13] Audun Jøsang. An algebra for assessing trust in certification chains. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS'99). The Internet Society*, 1999.

[14] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618 – 644, 2007. Emerging Issues in Collaborative Commerce.

[15] James Macfarlane. *Network Routing Basics: Understanding IP Routing in Cisco®Systems*. Wiley Publishing Inc, 2006.

[16] P. A. Nixon, W. Wagealla, C. English, and S. Terzis. *Security, Privacy and Trust Issues in Smart Environments*. Pearson Press, 2004. Chapter in Smart Environments, D. Cooke and S. Das (Eds).

[17] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation systems. *Commun. ACM*, 43(12):45–48, 2000.

[18] William Stallings. *Protect your privacy: a guide for PGP users*. Prentice-Hall, Inc., USA, 1995.

[19] William Stallings. *Cryptography and Network Security Principles and Practices*. Pearson Hall, USA, 5 edition, January 2010.

[20] John R. Vacca. *Public Key Infrastructure: Building Trusted Applications and Web Services*. Auerbach Publication, USA, 2004.

[21] Weiliang Zhao, Vijay Varadharajan, and George Bryan. Modelling Trust Relationships in Distributed Environments. In *TrustBus*, pages 40–49, 2004.

[22] Philip R. Zimmermann. *The Official PGP User's Guide*. MIT Press, USA, May 1995.

## Author Biographies

**Sarvjeet Herald** Born in India, Sarvjeet is a PhD student in the Algorithms Research Laboratory at the University of Hertfordshire. He was awarded a University prize for scoring "an unbroken string of A1s" in his M.Sc. (Distributed Systems and Networks) at the University of Hertfordshire (2009). His B.Sc. studies at St. Stephen's College (University of Delhi), India focused on different areas in Computer Science, Physics and Mathematics (2008). His research interest includes trust, security protocols, network security and cryptography.

**Stephen Clarke** Born in Chelmsford, UK, Stephen holds a BSc and PhD in Computer Science from the University of Hertfordshire, UK. His research and teaching interests are in security and trust in computer networks and distributed systems. Currently, he is a visiting lecturer at the School of Computer Science at UH.

**Bruce Christianson** New Zealander Bruce Christianson is Professor of Informatics at the University of Hertfordshire. Originally trained as a Functional Analyst, specializing in representation problems in Information and Communication Theory, he spent several years as a Consultant in the Communications Business Unit of Data Connection Ltd, before joining the University of Hertfordshire (then The Hatfield Polytechnic) in 1987. Bruce is widely known for his work on Optimistic Security for Open Distributed Systems.