Networking in The Solar Trust Model: Determining Optimal Trust Paths in a Decentralized Trust Network

Michael Clifford
The Aerospace Corporation
2350 East El Segundo Blvd.
El Segundo, CA 90245
clifford@aero.org

Abstract

The Solar Trust Model provides a method by which the sender of a message can be authenticated, and the level of trust that can be placed in the sender of the message or the message itself can be computed. The model works even if there is no prior relationship between the sender and receiver of the message. The Solar Trust Model overcomes a variety of limitations inherent in the design of other trust models and public key infrastructures. This paper presents a variety of enhancements and formalizations to the basic concepts of the model. In addition, this paper provides a set of algorithms that can be used to determine all of the possible trusted paths along which a message can be sent from a sender to recipient and the optimal choice of paths from a selection of paths. The paper also presents algorithms for reducing the network load produced by the model through piggybacking, path caching, and load distribution techniques.

1. Introduction

Electronic commerce requires both authentication and trust. For example, in order to make a purchase over the Web, authentication is required so that the purchaser of goods or services can be sure of with whom they are talking. Trust is required in order for the purchaser to feel confident that they will actually receive the goods that they purchased, and for the supplier to be able to believe that they will actually receive their money in exchange for their product. Likewise, in many other fields, it is critical that the recipient of a piece of information be able to tell both who generated that information and whether or not the information is likely to be accurate. For example, a patient who obtains medical advice must be confident not only that they are talking to a doctor, but that the doctor also has sufficient skill and experience to be able to diagnose their problem correctly. Traditional trust and PKI models offer either limited trust computation or authentication services, but not both.

Traditional models are also limited by a uniform notion of trust that assumes that all users of the model will regard trust in exactly the same way or will trust a single ultimate Trust Authority. Because user requirements for trust can vary significantly depending on the user and context, a truly functional trust model must be able to determine a level of trust that meets each user's unique requirements, and to adapt easily to changing conditions. It is not realistic to assign an exact numerical value to a level of trust or to use a universal formula to make such a computation, since this would force the user to accept either a judgment about a level of trust that they may not agree with, or a definition of trust that is imposed from outside. Rather, it only makes sense for the end user to determine an ordering of trust, and to make judgments about a particular situation relative to that ordering. This notion of trust is supported by [14].

The PGP/X.509 model [1] assumes an implicit transitivity of trust between participating parties. The PEM certification model [2] assumes that everyone in the world trusts one ultimate authority to verify the identities of other certificate senders. The PEM model also does not allow for multiple levels of trust within its certification hierarchy. Neither of these models provides a practical, universally applicable method of verifying the identity of the sender of a message. In the real world, trust is rarely transitive, and the concept of a single hierarchy has already given way to multiple corporate and governmental certifying authorities, each of which issues their certificates without being required to meet the standards of some ultimate certifying authority (CA) [3]. The Biba Integrity Model [4] provides for multiple levels of trust within an organization or

system, but not between two autonomous systems. The Common Security Services Manager's Trust Policy Interface Specification [5] provides an API in which an open set of authentication policies may be implemented, and permits multiple certificate authorities to sign the same message, but provides no method for verifying the trustworthiness of any of the CAs. The ICE-TEL trust model [6] eliminates the need for transitivity of trust, and permits certification along paths rather than in global CA hierarchies, but it does not provide for multiple levels of trust. The limitations on these models are discussed in great detail in [7], which introduced the concept of the Solar Trust Model.

The Solar Trust Model [7] enhances verification capabilities beyond other models by providing a method for designating many levels of trust, for permitting unlimited numbers of independent authenticators with no requirement for a central authority, and for determining the trustworthiness of a message that has been signed by an authenticator with whom the receiver of the message has no direct relationship. Each end user is able to judge on the fly a level of trust that can be assigned to an interaction, event, or other entity. These judgments are made relative to other previously made judgments, and are always made in a way which is appropriate to the context of the situation.

This paper expands on the work in [8,9,10] by describing new algorithms for the discovery of trusted paths, choosing optimal paths, and for reducing network load through caching and piggybacking methods. Additionally, a number of refinements and enhancements to the Solar Trust Model are presented, and the entire model is described in a more formal manner than has been done previously.

2. Background

The Solar Trust Model describes a method by which both the identity and the trustworthiness of the sender of the message can be determined, regardless of whether or not the message was signed by a certificate authority or other entity with which the recipient has a relationship. Because the Solar Trust Model operates in a manner different from other trust models, it is important to precisely define the terms and concepts that are used in the model.

A **Solar Trust Server (STS)** is the primary instrument of implementation of the Solar Trust Model. More precisely, the server is the combination of hardware and software that permits messages to be routed along specified paths, and that permits specific qualities about the message (in particular, trust) to be interpreted. **Trust** is the level of confidence that an STS places in the ability of another entity to meet certain criteria. As with any server, an STS has users. A **User** is anyone or anything that uses an STS to interpret the level of trust of a message, or to send and receive messages with the intent of following the rules of the Solar Trust Model.

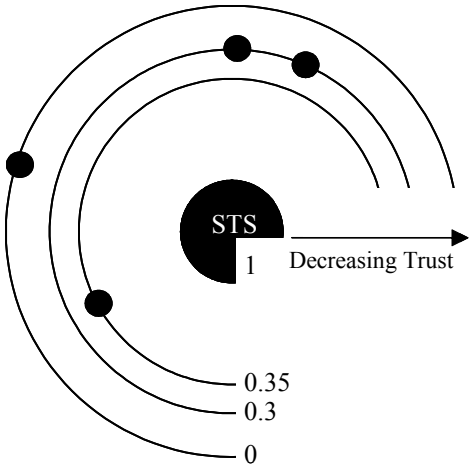

Figure 1. A typical Solar System. Four orbits have been defined in this example. Orbit 0.3 contains two entities. The other orbits contain one entity each. Orbit 1 contains the STS.

An **Entity** is any user, STS, or organization. Entities are grouped into orbits. An **Orbit** is an unordered set of entities. Orbits are labeled with rational numbers. Rational numbers are used for several reasons. First, there are an infinite number of rational numbers and therefore an infinite number of possible labels for orbits. Second, given any two rational numbers, it is always possible to find another rational number between them. Because it is always possible to label an orbit such that its label falls numerically between the labels of two other orbits, and because rational numbers can always be ordered by value, these labels provide a way to create a relative ordering of orbits. It is important to realize, however, that the label of an orbit is really just a name for the orbit, and that orbits do not behave in the same way that numbers do. For example, you cannot add or subtract orbits from one another.

The primary structure upon which the Solar Trust Model is based is called a **Solar System** (see Figure 1). A Solar System consists of an STS, and an ordered set of disjoint orbits. The orbits around an STS are ordered by level of trust. The most trusted (highest valued) orbit in a solar system is orbit 1.0. This orbit contains the STS, since the STS always trusts itself completely. The least trusted orbit in the ordering is designated as orbit 0.0. An entity in orbit 0.0 is considered to be completely untrustworthy.[1]

[1] In contrast with the original approach proposed for the Solar Trust Model in [7], the range of values that can be assigned to orbits in a solar system have been changed from integers that increased from 0 to infinity to rational numbers that decrease from 1.0 to 0.0. This has been done in order to simplify a future implementation.

It is easiest to conceptualize a solar system as analogous to a solar system in space. The solar system has a **Star**, or STS, at its center and is surrounded by concentric orbits that contain other entities. (The analogy to a solar system in space is not exact. For example, in a solar system in space, two objects do not generally share the same orbit, nor are the orbits necessarily concentric.) The set of all Solar Systems is called the **Universe**.

When a person or organization interacts with another entity, it may place some level of confidence or trust in that entity. For example, a person may place a certain level of trust in their friends, a different level of trust in their family, and a third level of trust in their colleagues. An STS is able to represent a level of trust through the use of a **Relationship**. A relationship is a function *f(object, solar system, policy) = orbit*. Given some object, a solar system and a policy, the relationship function maps the object into a specific orbit in that particular solar system using the policy. Relationships are not associative. For example, if STS A has a relationship with STS B, there is no guarantee that STS B has the same relationship, or any relationship all, with STS A.

Relationships come in different forms. For example, while a person may know their friends directly, they may only know the friends of their friends indirectly. For an STS, a **Direct Relationship** is any relationship that an STS has with another entity that the STS has defined using only its own policies and knowledge of that other entity, without relying upon the policies and knowledge of some intermediary third party. Such entities include other servers and users. In keeping with the solar system analogy, any entity with which an STS has established a direct relationship is known as a **Planet**. A **Direct Trust Relationship** is a direct relationship that is based upon the level of trust that an STS has in another entity.

One of the roles of an STS is to determine the level of trust that can be placed in a message, or in the sender of that message. A **Message** is a set of data that is sent from one STS to another. The contents of the message are arbitrary, and a message may contain other messages. When a message is transmitted, it is always sent along a predetermined **Path** (see figure 2). A path is an ordered set of pairs <S, O> such that the following conditions are met:

1) S is an STS.
2) For each S, O is the orbit in the solar system of S that contains the entity from which that S directly received the message.
3) The set contains all of the STSs traversed by the message.
4) The set is ordered by the sequence in which the message traversed those STSs, beginning with the STS that ultimately received the message, and terminating with the STS that originally sent the message.
5) If an STS appears on the path more than once, each instance in which it appears is considered to be distinct.

$S_{Receiving}$	$O_{Receiving}$
⋮	⋮
S_3	O_3
S_2	O_2
$S_{Sending}$	$O_{Sending}$

Figure 2. A Path

Paths are unidirectional. The fact that a path exists from A to B to C does not imply that a path exists from C to B to A. A **Subpath** is any path that can be composed by recursively taking a previously found path and removing the terminating pair in that path. (A path of length *n* has *n-1* subpaths.) An STS operates on a specific kind of path called a **Path of Trust**: This is a path where, with the exception of the original sending STS, all of the STSs in the set have a direct trust relationship with the next STS in the set.

Given that a message has been sent from an entity E to STS R_n through one or more intermediate STSs R_1 … R_{n-1}, an **Indirect Relationship** is any relationship that R_n has with E, where the relationship is based explicitly on the combination of policies of each individual STS along the path followed by a message (see figure 3). Any entity with which an STS does not have either a direct or indirect relationship is always considered to be in orbit 0 of that STS.

An **Indirect Trust Relationship** is any indirect relationship that is based explicitly on the combination of policies of each individual STS along a path of trust. STSs dynamically create, change and remove direct and indirect trust relationships with each other. This results in the formation of an ever-changing network of trust relationships.

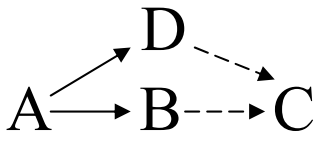

Figure 3. A has a direct relationship with B and D, and two different indirect relationships with C (one through D and one through B).

3. The Solar Trust Model and Ad Hoc Networks

A network of trust relationships between STSs behaves in a very similar manner to an **ad hoc wireless network**. In an ad hoc wireless network, the presence or absence of a unidirectional direct connection between two nodes depends on whether or not the two nodes are within range of each other. By extension, the presence or absence of a unidirectional indirect connection between two nodes depends on whether there exists some set of nodes such that there exists a path between the sending and receiving node where every intermediate node along the path is within range of both an upstream node and a downstream node. Additionally, because the nodes in an ad hoc wireless network can move around, turn off completely, or suddenly turn back on, a transmission path that was valid at one particular moment may not be valid the next moment.

A direct trust relationship from one STS to another can change at any time depending on the criteria that are used by the maintainers of the STS to determine what direct trust relationships the STS should have. Although a direct trust relationship may exist, the relationship may place the entity with which the relationship exists in such a low numbered orbit that the entity is not considered to be sufficiently trusted for a particular purpose. Likewise, the existence of an indirect trust relationship between two nodes depends on whether or not there exists some set of direct relationships between the receiving and sending nodes such that each receiver sufficiently trusts the next sender on the path. Because these relationships can change at any time, and because the policies that interpret these relationships can also change at any time, a path of trust that was valid at one particular moment may not be valid at the next moment.

Because of the similarity in structure between an ad hoc wireless network and a network of trusted paths and STSs, and because the problem of routing data in an ad hoc wireless network is well studied, this paper proposes to adapt the best routing techniques for ad hoc wireless networks to networks of STSs and their trust relationships. Dynamic Source Routing (DSR) was developed to address the problem of routing information on an ad hoc wireless network [8,9]. Performance simulations [11, 12] have demonstrated that the DSR protocol provides superior performance to other protocols for ad hoc networks. Additionally, various optimizations can be performed on the DSR protocol to further improve its performance [10]. This paper uses a modified version of the Dynamic Source Routing protocol, and incorporates both the ideas of using heuristics for load distribution [10], and route caching optimizations [13], that have recently been proposed for DSR. This paper also incorporates a variety of new optimizations that are specific to the Solar Trust Model, but which also might be applied in the future to the optimization of DSR.

4. Solar Trust Model Networking

4.1 Server Initialization

When an STS is first turned on, it has an empty solar system: every orbit is completely empty except for orbit one, which contains the STS itself. The administrator of the STS must establish a policy to assign meaning to one or more orbits and will label these orbits with an appropriate rational number between 0.0 and 1.0.

Direct trust relationships are then assigned to the STS. These may be assigned through some automatic process (for example, the STS might search a database for other STSs that meet some specific criteria), or entered manually. When an STS forms a direct relationship with another entity, the orbit in the STS's solar system that the entity is mapped to depends on the level of trust that the STS places in that entity relative to the levels of trust that have been assigned to the orbits of the STS's solar system. The relationship that an STS has with a particular entity may change over time, and may even be discontinued altogether.

4.2 Trusted Path Discovery

When an STS adds a new direct relationship with another STS, a set of indirect relationships with other STSs is implicitly established through that relationship. However, in order to determine exactly what indirect relationships exist, the STS must perform a path discovery. The idea of a path discovery is for an STS to learn all of the possible trusted paths that meet a certain set of criteria. The meaning of the criteria is up to the operator of the STS, however the acceptable range of values of the criteria is represented using one or more rules. A **Rule** is a function that outputs a binary decision based upon the entity that created it, and some criteria set by that entity. Given the following terms:

Rs = The relationship of the STS to the sender of the message, represented by the path that the message followed.
O = The orbit in which the entity from which the message was directly received is located.
P = The path taken by the message.

Rs(P) = The lowest labeled orbit O' for which a message arriving along path P can be trusted as much as a message originating directly from an entity in O'.
A rule is anything that satisfies these conditions.

Multiple rules can be applied together as a **Rule Set**. A rule set is an ordered list of rules. Each STS maintains a rule set. When an STS wants to transmit a rule set, it places the rule set into a **Rule Set Header** (see figure 4). A rule set header is a set of data that is attached to a message using a specific format. The header contains one or more

Field A: Local Direct Range
Field B: Local Indirect Range
Field C: Maximum Path Length
Field D: Permitted Local Range for Next CA

Figure 4. The Solar Trust Model Rule Set Header.

Rule Sets. The following fields are represented in the Rule Set Header:

- *Field A*: Local Direct Range: Trust all messages that have been directly signed by an STS in an orbit with a label that is greater than or equal to this value.
- *Field B*: Local Indirect Range: Trust messages that have been indirectly signed by an STS in an orbit with a label that is greater than or equal to this value. Do not trust any messages that have been signed by a STS in an orbit with a label below this value.
- *Field C*: Maximum Path Length: A message can be trusted only if the total number of STSs that have signed the message is no greater than this integer value.
- *Field D*: Permitted Local Range for Next STS: A message that has been indirectly signed by an STS inside the local indirect range can be trusted only if it came from an orbit in that STS's system with a label of no less than this value.

For example, let's say that Alice operates an STS, and that Alice does not consider any message that comes from an orbit lower than 0.8 to be sufficiently trustworthy for a particular purpose. Then Alice will set the value of Field A in the rule set for her STS to 0.8. Now let's say that Alice also trusts messages that reached her STS through an indirect relationship. She can specify a value of 0.9 in Field B to guarantee that only messages that arrive through indirect relationships that originate in an orbit with a label of 0.9 or greater will be trusted. If Alice enters a value in Field C, then a message sent to her STS will only be trusted if the length of the path taken by the message is less than or equal to this number. Finally, if Alice's STS has a direct relationship with other STSs, and she knows something about the criteria used by those STSs to determine their own trust orderings, then she may choose to restrict messages that come indirectly from those STSs to certain orbits within the Solar Systems of those STSs by specifying a value for field D.

Rule Sets from many STSs can be concatenated together into a **Composite Rule Set** (see figure 5). A composite rule set is a set of policies that as a group interpret the level of trust that the recipient of a message can place in the path followed by that message. A composite rule set is created by concatenating all of the rule sets for the STSs in the order that the message has passed though them.

Rule set of receiving STS (A)	Rule Set of STS B, with which A has a direct relationship	Rule Set of STS C, with which B has a direct relationship	Rule Set of Sending STS D, with which C has a direct relationship

Figure 5. A Composite Rule Set. The rule set for each STS that a message passes through is added on to the left side of the composite rule set. The complete composite rule set is read from left to right.

4.3 Path Discovery Algorithm

To perform a path discovery, a server sends out a Path Query Message to an STS with which it has a sufficiently trustworthy direct relationship. A path query message is composed of an ordered set of one or more 4-tuples of the form <S, O, N, T>, where S is the identity of the sender of a query, O is the orbit of S that the STS that the query is being forwarded to is in, N is the maximum path length from Field C of the Rule Set Header, and T is an integer representing time in seconds chosen by each S individually. It may also contain additional path queries attached to the end of the message (see figure 6).

S	O	N	T	Piggybacked path queries

Figure 6. Format of a Path Query Message.

A new query is normally initiated either at the time when that relationship is created, or when that relationship becomes sufficiently trustworthy as the result of a change in server policy or a change in the relationship. To minimize the flooding effect that can result from sending out a path query, a server with more than one new or changed direct relationship should space out the transmissions of path query messages to multiple STSs.

Path query messages are not sent to any STS that is not in a sufficiently trustworthy direct relationship, since any message that was received indirectly through that STS would be considered insufficiently trustworthy and would be discarded. When a path query message is received from another STS, it is saved in the STS's **Received Query Cache.**

Two other caches are initially used during path discovery. The first is an **Update Path Cache**. This is a cache of paths, and all of the subpaths of those paths, that an STS has learned of as a result of receiving them in path queries. These paths are used for performing path updates, and also represent possible ways in which the STS could *send* a message to a particular receiver. The set of all paths that the STS has already determined that it trusts is saved in the **Trusted Path Cache**. Put another way, the Trusted Path Cache contains all of the paths along which the STS can *receive* messages in a trusted manner.

Upon receiving a path query message, an STS executes the following algorithm:

1) Decrement the value of N in each 4-tuple in the query message.

2) If the STS has not created or forwarded a path query message recently, add a 4-tuple <S, O, N, T> for itself to the end of the query message.

3) While there exists a 4-tuple in the query message which contains an N with a value of zero:
    a. Define the head of the list as the ordered subset that begins with the 4-tuple at the beginning of the query message and ends with the first 4-tuple that is identified with a value of N = 0.
    b. Define H as the number of members (4-tuples) in the head of the list.
    c. Create a path by taking each pair <S, O> from the 4-tuples in the head of the list in order and adding it to the path.
    d. Save a copy of this path in the STS's Update Path Cache.
    e. Create a list of times T_1 … T_n by taking each T from the 4-tuple in the head of the list in order and adding it to the list of times.
    f. Generate RND = a random rational number between 0 and 1.
    g. While the path has a length > one:
        i. Wait for an amount of time $T_w = T_1*RND*H$
        ii. Send a copy of the path to the STS identified by the S in the pair at the head of the list. (The transmission should be made out of band, using a standard TCP connection for example.)
        iii. Remove the pair at the head of the list .
        iv. Remove the time T from the head of the list of times.
    h. Remove the head of the list from the path query message.

4) If part of the list in the query message remains, save this list in the current STS's Received Query Cache.

5) If the current STS's Local Direct range contains no planets
    a. Set the value of N in the 4-tuple at the end of the query message = 0.
    b. Repeat step 3.

6) Otherwise, for each STS that is a planet in the current STS's Local Direct range:
    a. Assign the value of the orbit that that planet is in to the O in the 4-tuple at the bottom of the list.
    b. Forward the query to that planet.

This algorithm does several things. First, the STS that initiates the path query will receive a set of paths that describe all sufficiently trusted paths to that STS. The paths that are transmitted to the STS will be of either maximum path length, or of the maximum length that the path could reach before another pair <S,O> could not be added to it. (This would happen because the last STSs to receive the path query did not have a direct relationship through which the indirect relationship could continue (see figure 7)). The STS that initiated the path query can then use these paths to find the set of all trusted paths to it, since any subpath of a sufficiently trusted path that ends at the STS must also be a sufficiently trusted path. All of these paths and subpaths are saved in the trusted path cache. Second, it allows any intermediate STS to piggyback its own path discovery query on top of the initial path discovery query if it needs to do so. Little additional network overhead is added by doing this because an STS that needs to do a path query would do so anyway, and piggybacking allows the STS to use path discovery information that was already propagating through the network (see figure 8). Note that in step 3g, paths of length 1 are not returned to query initiators, because a path of length one indicates a direct relationship, which the initiator must already know about by definition. Third, the time delay built in to the algorithm scales the response time to the query of each node that is sending a path back to the initiator of a query. Because the number of responses to a query can be expected to grow exponentially with each increase in maximum path length, there is a danger that a query initiator, and the physical network that it connects to, will be overloaded with a flood of responses. By setting an appropriate value of T, each query initiator can control the

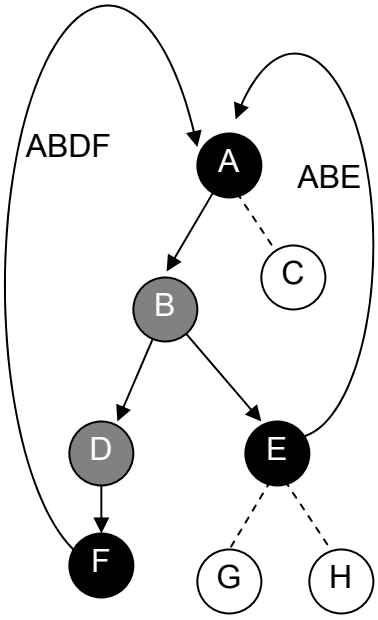

Figure 7. Propagation of a path query: Straight arrows indicate sufficiently trusted direct relationships. Dashed lines indicate insufficiently trusted direct relationships. STS A initiates a query with Maximum Path Length N = 4, and receives paths ABDF and ABE out of band (curved arrows).

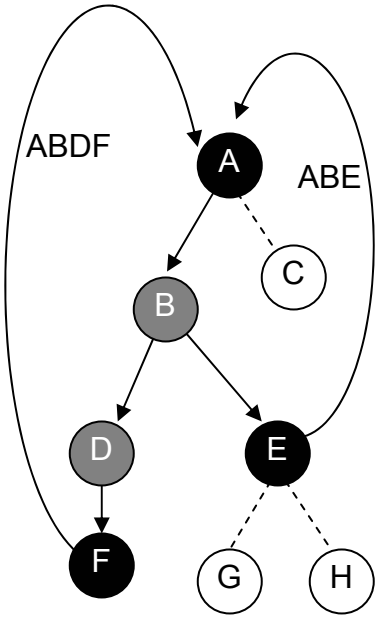

Figure 8. B has a Maximum Path Length of 2. It receives a path from a piggybacked path query from an STS two nodes away. STS A might otherwise have a longer acceptable path, but B's policy truncates that path. Queries added by D or F are not affected by B's maximum path length.

rate at which it receives responses. Furthermore, the use of an independent random number generation by each query responder ensures that the responses that are generated will be distributed evenly over a period of time that increases with the number of likely respondents. Finally, this algorithm guarantees that an STS that initiates a query will never receive a path that is not sufficiently trusted at the time the query passes along the path.

4.4 Path Sorting And Optimization

Once an STS has cached a set of trusted paths, it can determine which of the subset of paths from a particular sender is optimal according to a certain criteria. Since each server defines trust in its own way, each server will maintain a different policy about which orbit to map a particular path into. For a direct path to the STS, no mapping is necessary because the orbit in which a message originated is already known. For an indirect path to the STS, the composite rule set for that path is used by the STS in order to properly perform the mapping. One possible way for the STS to do this would be to start with the orbit from which the message was directly received as a baseline, and then to make adjustments as appropriate. In general, it is expected that the trust in a message that follows a particular path will degrade as the length of the path increases, and as the message passes through orbits of STSs solar systems that are increasingly far from the center of the solar systems.

One effect of performing such a mapping is that all of the paths of trust from a particular sender to the STS will automatically be ordered by relative trust. This makes it easy to compute an **Optimal Path.** An optimal path is a path that produces the most desirable result according to certain criteria when compared to other possible paths. The **Most Trusted Path** therefore is an optimal path, which when evaluated using the composite rule set for that path, will produce a greater level of trust than other paths of trust. Therefore, the most trusted path from a sender to a receiving STS would be the path in the orbit closest to the STS (with a label closest to 1.0). Similar methods can be used to determine a **Least Cost Path**, which is an Optimal Path that minimizes the money, amount of computation, or time that must be expended for a message to travel along a path that will place it into an acceptable orbit of the receiving STS. In some situations, an STS may wish to receive a message from a sender along the path that is optimized for one or more qualities. In other situations,

any valid path from the sender to the receiving STS may be acceptable.

4.5 Path Update Algorithm

As mentioned in the path discovery algorithm, each STS keeps a cache of all paths to it that it knows about. An STS learns of these paths during a path discovery when it receives a path query, and caches them for future reference. If an STS changes a direct relationship with a planet in its solar system, all of the paths that rely on that direct relationship are invalidated. When this occurs, the STS must perform a path discovery on that planet if the planet is still considered to be sufficiently trusted for a particular purpose. (If the planet is no longer considered sufficiently trusted, then there exist no paths through it that could be considered to be sufficiently trusted.) The STS performs this path discovery using the following algorithm:

1) The STS chooses any query from its Received Query Cache and forwards it to the planet with which it has established a new or different relationship.

2) For every other query in the Received Query Cache, the STS:
    a. Removes the pair at the end of the query list. (This is the pair that corresponds to the STS's own query. That query was sent out in the previous step.)
    b. Forwards the query to the planet with which it has established a new or different relationship.

This algorithm updates the indirect relationships of any STS that has discovered paths that run through this STS, as well as the relationships of the STS that has changed its policy or relationships. As an optimization, it may be helpful to set a "do not propagate" bit to notify STSs that receive the query that they should not add their own queries on to the update query unless it is absolutely necessary (i.e., they also need to perform an update and are ready to do so). This would prevent update queries from triggering a new query flood when one is not needed.

4.6 Sending a message

When one STS (the sender) needs to send a message to another STS (the receiver), it must know which path that message must follow in order to reach the receiver. The receiver does not care what path the message takes from the sender. Regardless of the path that the message takes, the receiver will still have to make a judgment about the level of trust that should be assigned to the message using the Rule Set Header. However, it is in the best interests of the sender for the path taken by the message to be as trusted as possible. The sender can determine this path in one of two ways:

1) If the sender has a path to the receiver in its Update Path Cache, and this path was received recently, then it is likely that this path is still valid. There is no guarantee that this path is any more or less trusted than any other path, but sending a message along this path will probably result in the receiver considering the message to be sufficiently trusted. The longer that this path remains in the cache, the less likely it is that this path will still be trusted.

2) If the sender does not wish to use a path in its Update Path Cache for any reason, the sender can send an out of band query (most likely using a TCP based message) to the receiver identifying itself and asking what path the message should take. The receiver can then reply with a suitable path, and the sender will send out the message along that path.

When a sender sends a message, that message is combined with an appropriate path to form a **Message Packet**. A message packet is a data structure consisting of a path followed by a signed block of data consisting of a rule set header, an orbit, and a message (see figure 9). The following algorithm is used when sending a message:

1) If a path is not attached to the front of the message (i.e. if the message is not encapsulated in a message packet), determine the appropriate path for the message to take.

2) If the current STS is not the receiver:
    a. Remove the STS from the path.
    b. Create a header composed of:
        i. The STS's rule set header.
        ii. The orbit from which the message was received. (In the case where the STS is the original sender of the message, if the message comes from that STS itself, the orbit must equal one since an STS must trust itself completely. If the STS is acting on behalf of a user with whom it has a relationship, the orbit is the same as the value of the relationship.)
        iii. The identity of the STS from which the message was received.
    c. Sign the combined message and rule set header. The signed message and rule set header are now considered to be the message in the message packet.
    d. Forward the message packet to the next STS on the path.

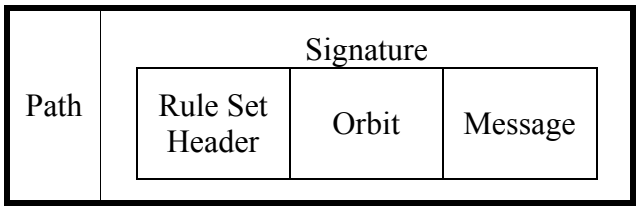

Figure 9. Format of a Message Packet.

4.7 Receiving and parsing a message

Once the message has reached the receiver, the receiver uses the information contained in the message to determine the level of trust that can be assigned to the message. The receiver builds a composite rule set header and a path by recursively reading the rule set, orbit, and identification information of each message, and verifying the message signature. The following algorithm is used to do this:

1) Begin a composite rule set by adding the rule set of the receiving server to an empty composite rule set.

2) Begin a path of trust by adding a pair <S, O> where S is the identity of the receiving server, and O is the orbit in the current server's solar system that contains the planet from which the message was directly received.

3) Do the following until the current message does not contain any smaller messages:
    a. Verify the signature on the message. If the signature is not valid, the message cannot be trusted. Stop processing on the message.
    b. Remove the rule set header from the message, and add it to the end of the current composite rule set for this message.
    c. Create a pair <S, O> where S is the server identity on the message and O is the orbit label on the message.

4) Add the pair to the end of the path of trust.

Once this process is completed, the server can make a judgment about whether, and by how much, the message should be trusted, by applying the rules in the composite rule set, in order, to the path that the message followed. Any message that is not sufficiently trusted will automatically be eliminated by the application of the composite rule set. The composite rule set is always parsed starting with the rules added by the receiving STS, and ending with the rules added by the sending STS. If a message is sufficiently trusted, the STS can make a judgment about the degree to which that message can be trusted by using its own policies to interpret the information contained in the path and the composite rule set.

4.8 Path cache updates

When an STS receives a message packet, it can use what it learns about the path contained in that message packet to update its path cache. If a message is deemed to be sufficiently trusted after following a path that is not in the path cache, that path is added to the path cache. Additionally, all subpaths from the receiver along the path to the sender can also be added to the path cache. If, on the other hand, the STS determines that the message followed a path that is not sufficiently trusted, and that the path that it followed was in the cache, then all paths in the cache that contain the untrusted path as a subpath must be deleted.

5. Direct Relationship Example

Imagine that Alice and Bob are surgeons in a hospital in Arlington. Alice's hospital runs an STS named Alpha. Based on the policy of the hospital, Alpha has placed Alice in orbit 0.85 and Bob in orbit 0.75 for trust issues relating to medical decisions.

Charlie is a surgical resident at the hospital. Dave, a patient with severe symptoms comes in to see Charlie. It is late at night, and both Alice and Bob are at home. The patient degrades quickly, leaving Charlie no choice but to perform an emergency surgery on the patient. A complication arises that exceeds Charlie's level of

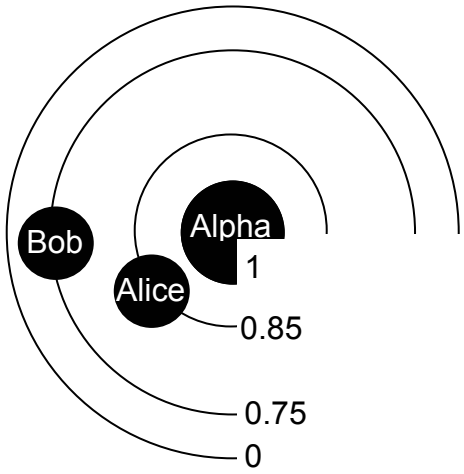

Figure 10. Alice's advice is considered to be more trustworthy than Bob's, because Alice is in an orbit with a higher numbered label.

expertise. Although Alice and Bob are both too far away to make it to the hospital in time to save the patient, Charlie can communicate with Alice and Bob through e-mail, using Alpha for authentication. He immediately sends a message to both of them asking how to deal with the complication. Alice replies that he must inject a drug to lower the patient's blood pressure. Bob replies that he must inject a drug to raise the patient's blood pressure. Following the wrong advice will certainly kill the patient, but which advice should he follow? Alpha assigns a greater level of trust to Alice's message then to Bob's message because Alice is in an orbit with a higher label than Bob. Therefore, Charlie chooses to follow Alice's advice. (See figure 10.)

6. Indirect Relationship Example

Several months later, Dave has moved to Denver when he has a relapse and must immediately have another surgery. Doris, Dave's surgeon in Denver, does not know Dave's medical history, and must obtain Dave's patient records from Arlington. She sends an out of band request for the records to Alpha through Delta, the STS of the Denver hospital. The Arlington hospital has a policy that describes the level of trust required for patient records to be sent to a third party. The Arlington hospital has no direct relationship with the Denver hospital, and therefore Delta is not in the solar system of Alpha. If the Arlington hospital's policy only permitted patient records to be sent to other hospitals or doctors with which it has a sufficiently strong direct relationship, then Alpha would not permit the patient records to be sent to Delta because Alpha does not have a direct relationship with Delta.

Fortunately, the Arlington hospital's policy also permits patient records to be transferred to other hospitals or doctors if a sufficiently trusted indirect relationship exists with that hospital or doctor. Alpha has a direct relationship labeled 0.6 with Bravo, the STS at the Boston hospital. Because the Boston hospital has worked closely with the Denver hospital in the past, Bravo has placed Delta in an orbit labeled 0.9. Based on the Denver hospital's own policy, Doris is in an orbit with a label of 0.638 in Delta's solar system. Alpha checks its trusted path cache, and determines that a path of trust to Delta has previously been computed using the path discovery algorithm. Furthermore, it also determines that a message sent along this path is likely to be sufficiently trusted to meet the requirements for the release of confidential patient records. Thus, Alpha replies out of band with the path from Delta to itself.

Delta creates a message packet with Doris' request, its relationship with Doris, and the appropriate rule set header, path header and signature. It forwards this message packet to Bravo, which repeats the process and forwards the message on to Alpha. Alpha determines that is the receiving STS. It creates a composite rule set and checks the signatures of Bravo and Delta. If the composite rule set says that Bravo and Delta are sufficiently trusted, and that Doris is in a sufficiently trusted orbit of Delta, then Alpha will permit Dave's patient records to be forwarded to the Denver hospital. Otherwise, Alpha will not forward the patient records and may choose to reply with a refusal.

7. Conclusion

The Solar Trust Model provides a method by which an entity can authenticate an entity with which it does not have a pre-existing relationship. It can also determine the extent to which it can trust that other entity, without the limitations inherent in traditional Public Key Infrastructures and trust models. A variety of enhancements to the model have been presented. The objects in the model have been better defined from [7]. A suite of algorithms has been presented for path discovery, path sorting and optimization, path updating, message receiving and path parsing. Some of these of algorithms are loosely based on the Dynamic Source Routing algorithm. These enhancements provide the missing elements that are necessary for the implementation of the Solar Trust Model. Implementation of the model could ultimately lead to a distributed, user-centric, and universal method of determining indirect trust relationships.

8. Future work

The use of additional optimizing strategies such as server capability based query and path routing are currently being investigated, as are additional path distribution mechanisms such as partial path advertisements. Future research directions include a simulation of these algorithms to determine their scalability, and to identify further optimizations that can be made to them. The implementation and testing of these algorithms in actual STSs is planned. Additionally, methods of preventing spoofing of query sender identities will be identified, in order to ensure that STSs can not be used for denial of service attacks.

References

[1] International Telecommunication Union. “Information Technology – Open Systems Interconnection – The Directory: Authentication Framework,” *ITU-T Recommendation X.509*, pages 7-17. 1995.

[2] Kent, S. “Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management.” Internet Report, RFC 1422, February 1993.

[3] L. Lpez, J. Carracedo. “Hierarchical Organization of Certification Authorities for Secure Environments.” Proceedings of the Symposium on Network and Distributed System Security, Februrary, 1997, San Diego, CA.

[4] Biba, K. “Integrity Considerations for Secure Computer Systems.” U.S. Air Force Electronic Systems Division Technical Report 760372, 1977.

[5] Intel Corporation. “Common Security Services Manager Trust Policy Interface (TPI) Specification Draft for Release 1.2”, page 7, March 1997.

[6] Young, A. Cicovic, N.K. and Chadwick, D. “Trust models in ICE-TEL”. Proceedings. 1997 Symposium on Network and Distributed System Security. (pp. 122-133), Feb. 1997.

[7] M. Clifford, C. Lavine, and M. Bishop, "The Solar Trust Model: Authentication Without Limitation," Proceedings of the 14th Annual Computer Security Applications Conference, 1998, pp. 300-307.

[8] Broch, J., Johnson, D., and Maltz, D. “The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks”. In Internet draft, IETF Mobile Ad Hoc Networking Working Group (Dec. 1998).

[9] David B. Johnson and David A. Maltz. “Dynamic Source Routing in Ad Hoc Wireless Networks”. In “Mobile Computing”, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153-181. Kluwer Academic Publishers, 1996.

[10] L. Wang, L. Zhang, O. Yang, Y. Shu and M. Dong, "Multipath Source Routing in Wireless Ad Hoc Networks", 2000 Canadian Conference on Electrical and Computer Engineering, Volume 1, p.479-483. 2000

[11] Meggers, J. and Filios, G., “Multicast communication in ‘ad hoc’ networks”. 48th IEEE Vehicular Technology Conference, Vol.1 (pp. 372 –376) 1998.

[12] Das, S.R., Perkins, C.E. and Royer, E.M “Performance comparison of two on-demand routing protocols for ad hoc networks”. IEEE INFOCOM 2000. Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Vol.1 Page(s): 3 –12, 2000.

[13] Marina, M.K.and Das, S.R. “Performance of route caching strategies in Dynamic Source Routing”. 2001 International Conference on Distributed Computing Systems Workshop, (pp. 425 – 432), 2001

[14] Bodeau, D. and Connolly, J. Findings from the May 2001 Workshop on Information-Security-System Rating and Ranking, as presented at the 17th Annual Computer Security Applications Conference in the tutorial on “Information Assurance ‘Metrics’”, Dec 11, 2001. http://www.acsac.org/2001/tutorials.html#tut7