# AttributeTrust – a Framework for Evaluating Trust in Aggregated Attributes via a Reputation System

Apurva Mohan and Douglas M. Blough
*School of Electrical and Computer Engineering*
*Georgia Institute of Technology, Atlanta, GA, USA*
*{apurva,doug.blough}@ece.gatech.edu*

## Abstract

*To enable a rich attribute-based authorization system, it is desirable that a large number of user attributes are available, possibly provided by multiple entities. The user may be required to aggregate his attributes and present them to a service provider to prove he has the right to access some service. In this paper, we present AttributeTrust – a policy-based privacy enhanced framework for aggregating user attributes and evaluating confidence in these attributes. We envision a future where attribute providers will be commonplace and service providers will face the problem of choosing one among multiple attribute providers that can provide the same user attribute. In AttributeTrust, we address this problem by means of a reputation system model based on transitive trust. Entities express confidence in other entities to supply trusted attributes, forming chains from a service provider to different attribute providers. A service provider uses this transitive reputation to decide whether to accept a particular attribute from a specific attribute provider.We discuss how the AttributeTrust model prevents common attacks on reputation systems. AttributeTrust differs from the current approaches by deriving its attack resistance from its specific context of attribute provisioning, its voting mechanism formulation, and unique properties of its confidence relationships.*

## 1. Introduction

Attribute-based systems provide a highly granular, scalable and semantically rich method for access control and authorization [8]. A service provider (SP) defines policies, which dictate the set of attributes required for accessing its services. The user provides credentials that contain verifiable attributes to gain access to these services. Different levels of access can be provided based on what attribute values the user possesses. This scheme is especially suitable for distributed systems, where maintaining real time synchronization among access control lists is a huge problem [7]. Another merit of this access control model is that, it does not require *a priori* knowledge about the user at the service provider.

Although attribute-based systems have many advantages compared to traditional systems like role based access control [8], they have some practical limitations. We examine some of these limitations as a way to motivate AttributeTrust. Firstly, it is very difficult for a service provider to directly verify each user's attributes in real time. Traditionally, the problem of verification is addressed by requiring users to get their attributes bundled into credentials, verified and digitally signed by an Identity Provider (IdP) [3, 4]. These credentials are trusted by the SP (which acts as a relying party). We argue that this approach limits the attribute set that can be used because IdPs might not be willing to certify certain attributes or it might not be efficient for them to verify these attributes. For example, the IdPs may be unwilling to certify attributes that are not identity related. Another problem is with rapidly changing attributes or adding new attributes. Fresh credentials need to be issued by the IdP every time an attribute value changes or a new attribute is added.

Secondly, the abovementioned scheme also requires that all relying parties have reliable access to all the IdPs' public keys [5]. This scheme can either have only a few IdPs effectively overloading them and making them lucrative targets for attacks or have a high number of IdPs, making it difficult for each relying party (RP) to know every IdP's public key reliably [5, 9]. We argue that this represents a performance bottleneck in current systems.

Thirdly, in current systems, only a few user attributes are available. In order to achieve a high granularity in attribute-based systems, a large number of attributes are desired. Since an attribute is any parameter that characterizes a user, a single IdP may not be able to verify every attribute and issue credentials. If there are multiple IdPs, the user may be required to aggregate these attributes before presenting them to the SP [1, 2].

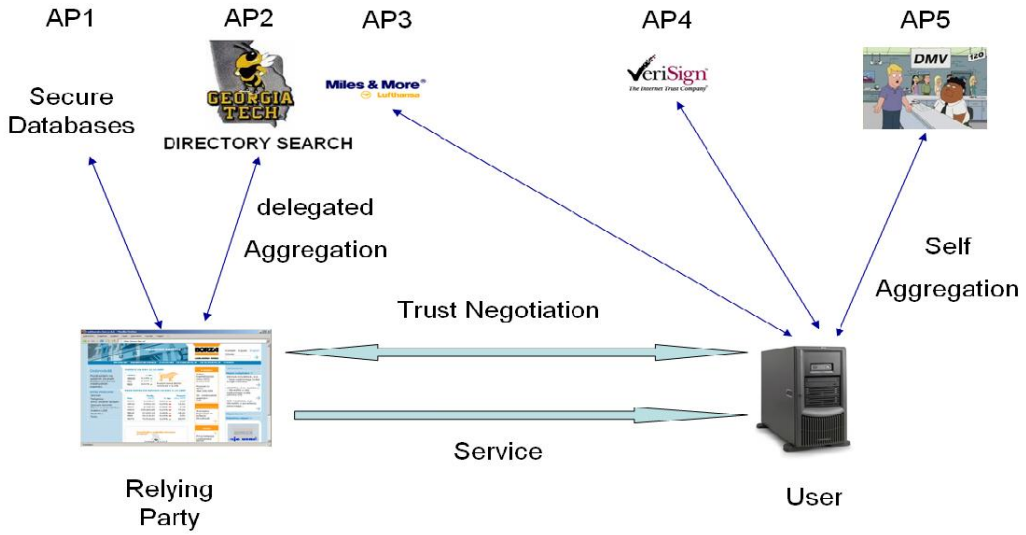


**Figure 1 – AttributeTrust Framework**

Fourthly, most current systems use a group of attributes contained in a credential for supplying user attributes. AttributeTrust provides a higher granularity by dealing with individual attributes. This enhances user privacy by limiting the unnecessary disclosure of attribute values.

Finally, we discuss the problem of how much trust a RP should place in attributes. This problem is separate from verifying the credentials themselves. IdPs have different certification policies, making the 'attribute verification strength' vary from one IdP to another. If all credentials are trusted equally, malicious users would get their credentials issued by IdPs with lenient certification policies with little verification. They may also try to subvert the system by registering as an IdP themselves and issuing bogus credentials to other malicious users. This problem is illustrated by an example. Alice's local social club, her university and the state department of driver services can each act as an IdP and issue credentials certifying Alice's date of birth. The RP can verify all the three credentials using the respective IdP's public keys but the question is *which IdP can the RP trust to provide this particular attribute*?

**AttributeTrust** – This paper presents AttributeTrust, a framework for aggregating user attributes, performing policy based trust negotiations, and evaluating trust in attributes. To address the problem of verifying a large number of attributes by the IdP, we introduce the concept of an Attribute Provider (AP), by which we mean something different from some prior usages of this term [22]. Our definition of an AP is "*an entity that holds user attributes and can provide them to the RP in a reliable way*". Traditionally, the functionality of the IdP is limited to providing signed user credentials, whereas the AP can operate in multiple ways to provide attributes reliably. First, the AP may provide signed credentials to the user in which case its functionality is similar to an IdP. Second, the AP may operate a public database holding user attributes; the user first proves her identity to the SP and points to her attributes in the AP's public database. Since the AP server is a trusted entity, attributes may be fetched from the server in a reliable way. Finally, the AP may store users' attributes in a secure database. The user provides the SP with a cryptographic token to be presented to the AP to release the attributes. The AP provides enhanced functionality compared to a traditional IdP. We discuss the advantages of the AP's enhanced functionality in Section 2.

Figure 1 shows the basic components of the framework. Entities are the User, the Relying Party and multiple Attribute Providers. The user requests some service from the RP. The RP initiates a trust negotiation, where the user and RP both are required to provide attributes based on the other party's disclosure policies. To provide these attributes the user may aggregate some or all these attributes from the APs himself (referred to as self aggregation) or provide a token to the RP to fetch some attributes from the AP directly (referred to as delegated aggregation). The RP also provides its attributes to the user, which may be aggregated. The RP and the user accept the other party's attributes based on their trustworthiness (the method for evaluating the trustworthiness is explained in the next paragraph). By providing the attributes, the RP and the user satisfy the other party's disclosure policies, thus completing the negotiation successfully. When the trust negotiation is successful, the RP provides the service to the user.

It is important to distinguish between trust in the paths leading to the attributes vs. trust in attributes themselves. In AttributeTrust, each AP's public key is stored such that they are readily accessible to all the entities, so the attributes can be verified unambiguously by the RP. On the other hand, the RP's trust in attributes provided by an AP is calculated by using a reputation system model. The system is represented as a graph where the nodes represent entities that may be users, RPs or APs and the directed edges represent the confidence the source node has in the destination node's ability to provide trusted attributes. When the RP wants to calculate its trust in attributes provided by an AP, it calculates a confidence value based on all the confidence paths leading from itself to the AP. If the RP does not directly trust the AP, it may have a trusted peer who has trust in attributes provided by the AP. In this way, chain of trust helps in calculating the RP's transitive trust in the AP. Even if the RP has a direct trust relationship with the AP, it is better to calculate a final trust value based on all the available confidence paths, so that the RP uses his and his peers collective trust to calculate his final trust. This final trust value can be used by the RP to determine if it can accept the attribute from this AP. The primary advantage of this scheme is that any entity can serve as an AP in the system; the RP will accept attributes from it only if the AP is in the RP's 'circle of trust'[1]. The details of the model are presented in Section 3.

We have studied some common attacks on reputation systems and made the model robust against such attacks. These attack resistant properties are discussed in Section 4.

The main contributions of the paper are – defining the scope of an AP's functionality, providing a framework for aggregating attributes from APs for use in trust negotiations and building an attack resistant reputation system for evaluating trust in aggregated attributes. The rest of the paper is organized as follows – Section 2 discusses attribute aggregation and its use in trust negotiations. Section 3 presents details of AttributeTrust, while Section 4 studies the attack resistant properties of AttributeTrust. Section 5 discusses how AttributeTrust satisfies some of the model requirements noted in Section 3.1. Section 6 discusses related work, and finally Section 7 concludes.

# 2. Attribute Aggregation and Trust Negotiation

In this section, we discuss the concept of Attribute Provider (AP), methods of aggregating attributes from AP's, and how to use these attributes in automated trust negotiations.

1. Circle of Trust means all the nodes covered in a circle with RP at the center and the radius is equal to the maximum path length (L)

## 2.1. Attribute Provider

In Section 1, we noted that the AP is an entity that holds user attributes and can provide them to the RP in a reliable way. We also presented a number of ways in which an AP can function. In this way, the AP encompasses some of the functionality of an IdP and also extends it in a way that allows any entity to set up an attribute server and register as an AP in the system. This is useful because now an AP provides attributes with which it is closely related (hence easily verifiable) and saves the IdP the trouble of verifying obscure (from the IdP's perspective) attributes. For example, whether Alice, who is a music major, is an expert or amateur piano player is something that can be asserted by her music school but it might be difficult for a traditional IdP to assess. This attribute may be of interest to a high school looking for a piano teacher. The main advantage of using this system is that a large variety of attributes can be provided and it can be applied in several new dimensions rather than just proving a user's identity. Whether the music school is qualified to provide this attribute is decided by the reputation system discussed in Section 3.

## 2.2. Attribute Aggregation

Any user in this system will usually have a number of attribute providers. While building mutual trust with a RP, the user is expected to provide a number of attributes [2]. These attributes need to be aggregated from APs. Depending on the APs' functionality, this may be done by the user or delegated to the RP by the user.

### 2.2.1. User mediated aggregation

This case applies when the AP issues signed credentials to the user. The user can have pre-issued credentials or can request freshly signed credentials. Minimum information disclosure can be applied where the AP can provide credentials with the attribute values hashed by a one-way function [19]. The user can provide the relevant attributes to the RP in plaintext, which can be verified by the RP by hashing them with the same one-way function and comparing against the signed credential [19, 20].

### 2.2.2. Relying Party mediated aggregation

The relying party can be delegated the task of aggregation by the user. This can be done in two ways. First, if the AP runs a public database, the user points the RP to a particular pseudonym's attributes in the AP's database. The user can establish ownership of the pseudonym in multiple ways like providing an identity

certificate, through previous relationship with the RP, or correlating other data the RP maintains about the user.

Second, the AP may be running a database answering queries from authenticated users. In this case, the user can delegate the aggregation by providing a signed cryptographic token to the RP, which can be presented to the AP for authentication and requesting the AP to release particular attributes.

One way to construct such a cryptographic token is discussed here. The user and AP set up a shared key. The user creates the token by specifying a particular set of attributes and encrypting them with the shared key. The advantage of using shared key is that this key can be changed independently for each AP, in case of any security breach at the AP.

In our example, the music school is the AP and Alice is the user. Alice creates a delegation token with key $K_{ALICE}$ for the attribute – 'PianoPlayer' as shown –

Token = {E(Attribute=PianoPlayer, $K_{ALICE}$), ALICE }

## 2.3. Trust Negotiation

When the user contacts the relying party requesting a service, the RP initiates an automated trust negotiation based on policies. The RP and the user both have attribute disclosure policies, meaning the disclosure of sensitive attributes are protected by policies. In a particular scenario, suppose the RP asks the user for a sensitive attribute. The user may have a policy that requires that the RP first prove its identity before the attribute can be released. For example, when the high school requests Alice to prove that she is an expert piano player, Alice may have a policy in place that requires the high school to prove its accreditation by the competent authority. These disclosure policies are configurable and can be dynamically updated. The user and the RP each have a negotiation policy tree similar to the one shown in Figure 2.

In a negotiation tree of depth n, an attribute at level k can only be disclosed when all the required attributes from level n up to level k+1 have been presented by the other negotiating party.

The policy tree shown in Figure 2 is an 'AND-OR' tree defining the release requirements for attributes. These requirements are expressed as AND (^) and OR (v) relations among the nodes at the next level in the tree. The OR relations are shown by dashed lines whereas the AND relations are shown by solid lines. Both the trees at the user and the RP side grow dynamically when each party learns about the other party's release policies for that attribute. For example, for releasing $A_{U1}$ the user may require that either $A_{R1}$ v $A_{R2}$ ^ $A_{R3}$ must be released by the RP. The RP in turn requires that the user release other attributes before it can release attribute $A_{R1}$ or attribute $A_{R3}$. A requirement at a node is satisfied when

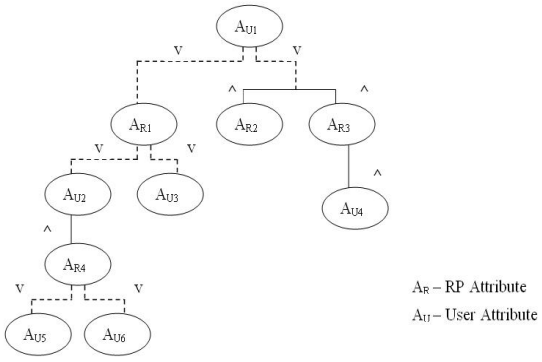


**Figure 2 – Attribute Release Policy Tree**

the policies of all its children are satisfied. The release policy of the root node can be satisfied if a path from any leaf node to the root node can be found where the policies for all the intermediate nodes as fulfilled. Only then the negotiation is successful. If no such path can be found, the negotiation is unsuccessful.

In case of circular dependencies in the policies, a proof of holding the attribute can be used to break the deadlock. For pre-signed credentials, this can be done by hashing out the attribute values in the credentials and presenting the credentials [20]. We extend this case for the AP. Pre-signed credentials can be used as described above. If the AP hosts a public database, the user can point to his entry in the database. In case of protected databases, the user can create a token requesting the AP to assert that the user holds the required attribute and present this token to the SP. SP can present this token to the AP to receive this assertion.

# 3. AttributeTrust

In this section, we present the AttributeTrust framework for measuring confidence in aggregated attributes. We use a reputation system model, where each member in the system votes for his peer's ability to provide trusted attributes. In Section 3.1, we discuss some desirable properties of the reputation system model. In Section 3.2, we show the basic formulation of the proposed model. Section 3.3 discusses the system implementation of AttributeTrust and finally, Section 3.4 presents the calculation of confidence metrics.

## 3.1. Desirable Properties

Before we present the model, it is important to understand some of the properties that are desirable in such a model. These properties will serve as a guideline as to how AttributeTrust is expected to behave.

i) Each entity should be able to compute confidence values using the model in reasonable time.
ii) The output of the model should be a metric that can be directly used in making authorization decisions.

iii) Evaluating the metric with only partial information should still give meaningful results (albeit with a lesser accuracy).
iv) The model should be robust against common attacks on reputation systems.
v) The metric should degrade gracefully in the event of an intensified attack.
vi) There should be disincentive for an honest user to abuse the reputation system.

### 3.2. Model Formulation

The system can be viewed as a weighted directed graph $G = (V, E)$, such that $E \subseteq [V]^2$. Each vertex (node) V represents an entity; every edge E represents a confidence relationship. The entities in the system can be users, APs or RPs. The edges are directed and weights on the edges are not necessarily symmetric i.e. for two entities a and b, a's confidence in b can be higher or lower then b's confidence in a. Establishing this directed edge relation is optional for the entities. Nodes in the system representing RPs and users are weighted and the node weight calculation is explained later in this section. Nodes representing the APs do not have a node weight. The order of graph |G| is defined by the number of entities in the system. G is irregular and incomplete. Edges and nodes are created dynamically in the graph, so |E| and |V| increases with time but for calculating the node and edge weights (to construct a path as explained in Section 3.4), we will consider a snapshot of the graph, which is static.

The node weight is represented by $\omega$. The node weight $\omega_i$ for node i is defined as –

$\omega_i = d_i * \overline{C}_i$ ----- Equation 1

$\overline{C}_i = \sum_{k=1}^{n} C_{k->i}$, where n is the number of incoming neighbors of node i ---- Equation 2

$d_i$ is the in-degree of node i

The node weight is a product of the in-degree and the average confidence value on the edges, which is simply equal to the sum of the confidence values. A high in-degree means a lot of nodes have expressed confidence in node i. A high average confidence value means that node i's peers have high confidence in it. So the node weight represents the community's confidence in node i. The edge relation value $C_{k->i}$ represents the confidence node k expresses in node i. It is a non-binary value between [0,1]. When an entity enters the system initially, its node and edge weight are zero because it is not connected to any other entity. The node weight increases when edge relationships are established. A default node weight of zero is helpful in reducing the impact of certain types of attacks. This is discussed in more detail in Section 4.

The edge relation between nodes representing an AP and a user is different from what is described above. These edges are always directed from the user to the AP. Instead of a single confidence value, the weight of this edge is an array where each array element represents a confidence value for an individual attribute.

### 3.3. System Implementation

In the system, each entity has a unique pseudonym. Each entity has a public-private key pair. The public keys of the APs are stored in the system in such a way that any entity can access them[2]. Entities interact with each other in the process of acquiring or providing service. They provide feedback in the form of confidence scores. The node weight is proportional to the in-degree. If an entity wants to provide feedback to its peer entity after a transaction, it should fulfill two requirements – i) have an irrefutable proof that the transaction occurred and ii) have a node weight greater than the participation threshold $\pi$. The participation threshold forces the entity to behave in an honest manner for some time and earn its 'reputation' before it can participate in the feedback process. On the other hand, to establish a user -> AP relation edge the user should satisfy requirement iii) instead of requirement i); iii) either the user has attributes certified by the AP or the user has performed a transaction in which it accepted attributes certified by the AP.

The weight of a node is dependent only on the edges terminating in it, so the weight of each node can be calculated independently. This helps in eliminating oscillations in weight calculation. These weights can be calculated in multiple ways. A central server can be the easiest implementation but it also makes the central server a lucrative target for attacks. A performance problem can arise by a very large number of nodes simultaneously querying the server for node weights. Alternately, a group of servers can be used to calculate the node weights. Each server calculates the weights for one group of entities with overlap, so that each entity's node weight is calculated by more than one server. This provides redundancy to resist failures and attacks. As another alternative, entities can calculate node weights where each entity is responsible for a small group of nodes and each node's weight is calculated by multiple entities [13].

---

2. In the case of an IdP's identity, knowing its public key reliably is equivalent to trusting the IdP. Trust brokering is performed by the CA through a complex trust hierarchy; hence establishing trust through trust chains is difficult. In the case of an AP, knowing the public key of an AP (trust brokered by the AttributeTrust system) is different from trusting the AP (which is based on the AP's reputation). Since there is a single trust brokering entity, it is easy to put all the AP's public keys at one place, the AP's still have to earn a high reputation to be trusted.

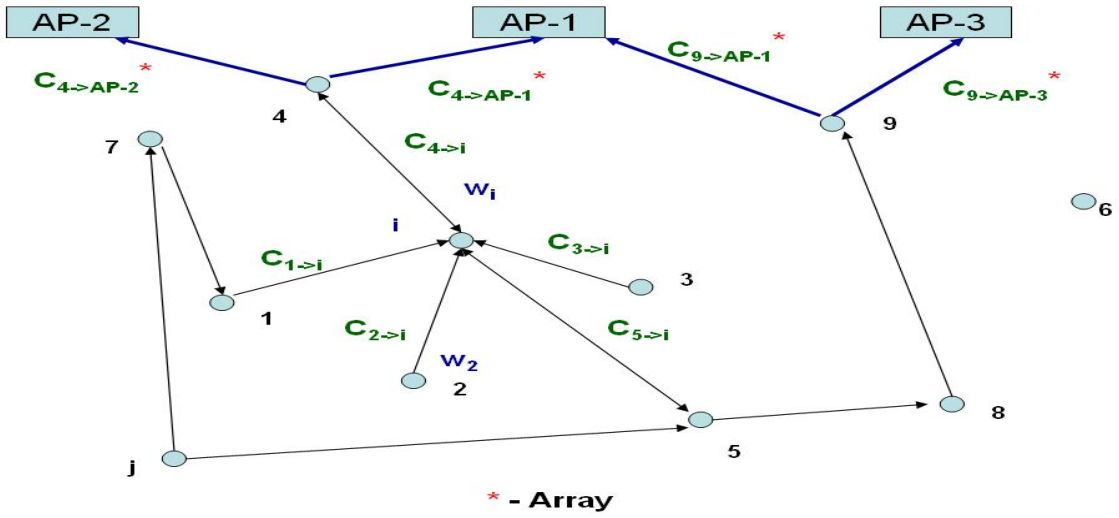


**Figure 3 – System Represented as Graph**

The edge weights are stored locally with each entity and used during the construction of confidence paths (explained in Section 3.4).

### 3.4. Metric Calculation

Referring to Figure 3, consider a scenario where node j is a service provider (acting as a relying party) and node i wants to acquire a service. During their negotiation, node i provides some attribute issued by AP1 to node j. In order to calculate its confidence in this attribute, node j finds all the paths from itself to AP1 of maximum length L. Using this parameter, the relying party places an upper limit to the number of redirections in the transitive trust chain. The Relying Party may not be comfortable accepting attributes from very far-off nodes, hence this parameter is useful. This parameter also helps the algorithm to converge in reasonable time. Node j asks all its peers if they have a confidence edge leading to AP1. The peers in turn repeat this process until either they find a path to AP1 or the path length exceeds L. j finds two paths viz j->i->4->AP1 and j->5->8->9->AP1.

Upon receiving values along all the paths, node j converts them into node disjoint paths. For paths which have parallel sub-paths, final confidence value is calculated (as explained later in this section) for each sub-path. The sub-path with the highest value is considered. For example, in Figure 4 we calculate confidence values of j->7->1->i and j->2->i and choose the higher value of the two.

For calculating the final confidence values, we define two functions – concatenation and aggregation.

**Concatenation ($\Theta$)** is used to find the confidence for one complete trust path. We multiply the confidence $C_{t->t+1}$ by $\omega_t$ and repeat the process for each of the nodes in the path (t represents intermediate node). For this, we consider $\omega_j$ as 1, instead of its global node weight, because j will have complete faith in the confidence value he gave to his peer.

$$\Theta = \omega_t * C_{t->t+1} \quad \text{-----} \quad \text{Equation 3}$$

**Aggregation** ( $\Sigma$ ) is used to find the final confidence value ($\Gamma$) considering all the node disjoint paths. There are many aggregation functions possible. We present four popularly used aggregation strategies and examine the effect of a slandering attack on AttributeTrust for these strategies in Section 4.4.

i) A conservative strategy is to choose the confidence value of the path with the minimum value.
ii) Another strategy is to find the average confidence value of all the available paths.
iii) Another popular strategy requires that at least 3 confidence paths are available. They are sorted in order of their confidence values and the median confidence value is chosen.

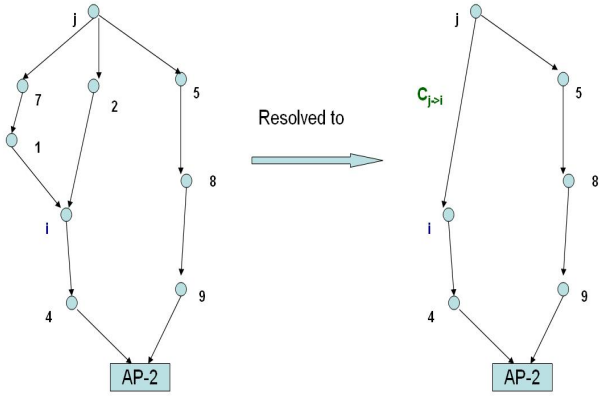


**Figure 4 – Node-joint Path Resolution**

iv) In the last strategy, known as the additive strategy, the final confidence value is the sum of the confidence values of all the paths; each newly created path will increase the final confidence value.

For each node disjoint path from node j to AP1, we apply the concatenation function to calculate the path wise confidence values and then we apply the aggregation function including all the paths to calculate the final confidence value $\Gamma$. The AP is considered sufficiently trusted by the RP if the value of $\Gamma$ is greater than some user defined threshold. A suitable default value will be set but the user will have an option to change this value depending on the sensitivity of the transaction. From the usability point of view, the system should have a default threshold value in place and the user should interact with the system only while providing feedback to its peer entity. To further improve usability, this feedback can be represented as several discrete levels like completely trusted, highly trusted, average, marginally trusted and untrusted.

## 4. Attack Resistance

In this section, we take a look at some of the common attacks and abuses against reputation systems and how the proposed model defends against them.

### 4.1. Whitewashing

When the reputation of a user in the system becomes very low, he discards his pseudonym and reenters the system with a new pseudonym. Although acquiring a pseudonym in the system is free, we provide a disincentive for the user to do that. When a new user enters the system, he is not connected with any other node and his node weight is zero. Low reputation equates to a low node weight in the system. If a user re-enters the system, he has a zero weight which is the minimum, so there is little incentive for him to acquire a new pseudonym.

### 4.2. Discrimination

Some users behave well with most of the users but discriminate against some of them. We address this problem by using a link drop threshold $\mu$. Suppose user i constantly discriminates against user j, the confidence value $C_{j->i}$ will keep on dropping. We define a threshold below which user j will not consider paths via user i for calculating confidence metric irrespective of $\omega_i$. This way user j can eliminate discriminating nodes from his confidence metric calculation.

### 4.3. Traitors

In this attack, a malicious user behaves honestly for some time and then tries to milk his reputation by cheating. This reputation building and milking cycle continues. During the reputation milking phase, the user cheats on a number of transactions continuously until his reputation is very low; and then either enters the reputation building stage or reenters the system with a new pseudonym. As an extension, AttributeTrust can provide a user's history of recent transactions, which the other party may use to call off a negotiation if the user had a majority of his recent transactions failed. This way AttributeTrust cuts short the milking stage of a traitor.

### 4.4. Slandering

A malicious user may try to malign the reputation of a reputed AP by launching a slandering attack against the AP. To launch this attack, the malicious user M creates a relation edge from himself to AP1 as shown in Figure 5. He can easily create this edge by performing a transaction with a user and accepting attributes from AP1. M then gives arbitrarily low confidence values to attributes provided by AP1. The effect of a slandering attack by M by adding a malicious path $P_M$ (to the honest paths $P_4$ and $P_9$) is shown in Figure 6. Similarly a group of colluding users can create numerous paths from user to AP1. The effect of this depends on the choice of aggregation strategy (different strategies presented in Section 3.4) and a comparison of different strategies is presented more formally below.

The aim of adversaries in this case is to reduce the AP's reputation by as much as possible. They try to achieve this by creating paths of very low trust values. Here, we present the cost of launching such an attack and a lower bound on the confidence value for different aggregation schemes. Let, $m$ and $\rho$ be the minimum number and actual number of paths respectively, (assuming, $\rho \geq m$), T be the confidence value of a path i, $\varepsilon$ be the confidence value of one malicious path, $\ell$ be the lower bound on the confidence value in the presence of malicious entities and $\kappa$ be the number of malicious paths created. Further, let, $\zeta$ be the cost of launching the attack and $\chi$ denote the cost of an edge attack[3], as described in [23]. Note: $\varepsilon \cong 0$ *s.t.* $\kappa.\varepsilon \approx \varepsilon$.

In the following, i to iv refer to the aggregation strategies discusses in Section 3.4.

**Theorem 1** *Using strategy (i), the final confidence value can be made arbitrarily low with a cost of* $\chi$.

***Proof***. If the malicious entity creates a single malicious path with confidence value $\varepsilon$, then $\ell = \varepsilon$ and $\zeta = \chi$.

**Theorem 2** *Using strategy (ii), the malicious entity's ability to bring down the final confidence value*

3. The cost of launching an edge attack is assumed to be uniform across all nodes as in [15] and [23].

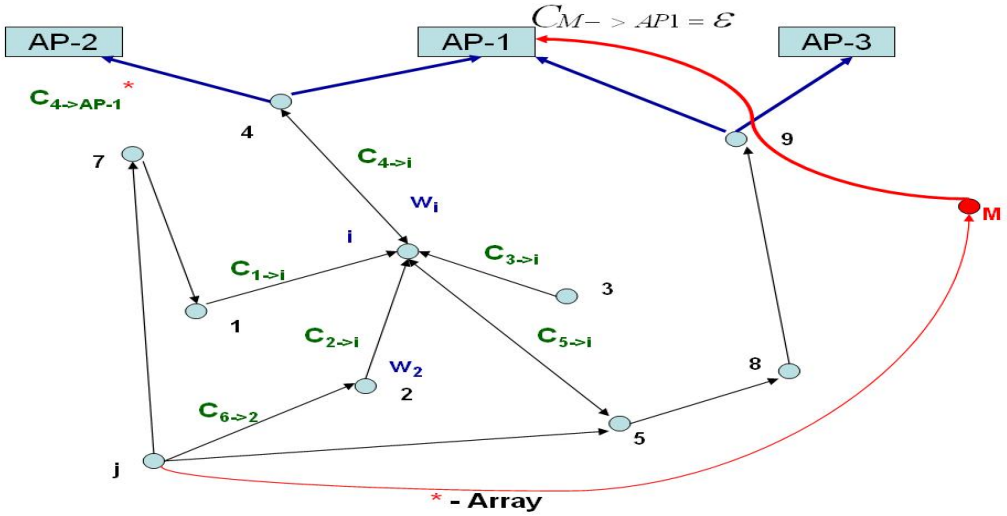


**Figure 5 – Slandering Attack**

| | | |
|---|---|---|
| Minimum Path – | $P_M$ | (High Impact) |
| Average of Paths - | $(P_4 + P_9 + P_M)/3$ | (Medium Impact) |
| Discard Extremes – | $P_9$ if $P_4 > P_9 > P_M$ | (Low Impact) |
| Summation over paths - | $(P_4 + P_9 + P_M)$ | (No Impact) |

**Figure 6 – Effect of Slandering attack on Aggregation Strategies**

*is limited by its ability to successfully attack disconnected nodes i.e. by the value of* $\kappa$.

***Proof***. To find the average, the summation of the confidence value of paths remain almost constant ($\kappa.\varepsilon \approx \varepsilon$) but the final confidence value decreases with increasing $\kappa$, $\sum_{i=1}^{\rho} T_i = P$, $\ell = \frac{P+\varepsilon}{\rho+\kappa}$ and the cost is $\zeta=\kappa.\chi$.

**Theorem 3** *Using strategy (iii), the final confidence value can be brought arbitrarily low if* $\kappa > \rho$. *If* $\kappa < \rho$, *then the final confidence value is at least* $P_l$, *where* $P_l$ *is the confidence value of lowest value honest path be given as* $P_l = \sum_{lowest-value-path} T$ .

***Proof***. If $\kappa < \rho$, then in the worse case, the median path is the lowest value path and $\ell = P_l$. On the other hand, if $\kappa > \rho$ then the median path will be one of the malicious paths and $\ell = \varepsilon$. The cost in either case is $\zeta = \kappa.\chi$.

**Theorem 4** *Using strategy (iv), the malicious entities cannot lower the final confidence value irrespective of their ability to perform edge attacks.*

***Proof***. In the additive strategy, each new confidence path can only increase the final confidence value, so for $\kappa$ malicious paths, $\ell = P + \varepsilon$ and the cost is $\zeta = \kappa.\chi$. Note that $P + \varepsilon > P$.

### 4.5. Self-promotion via Sybil Attack

As an advanced attack, M can create a malicious attribute provider AP-M, create a number of Sybil nodes [11] and try to increase the confidence value of AP-M directly by asserting high confidence values to attributes provided by AP-M. M can further increase AP-M's confidence indirectly by asserting high confidence values to AP-M through all the Sybil nodes. We use a

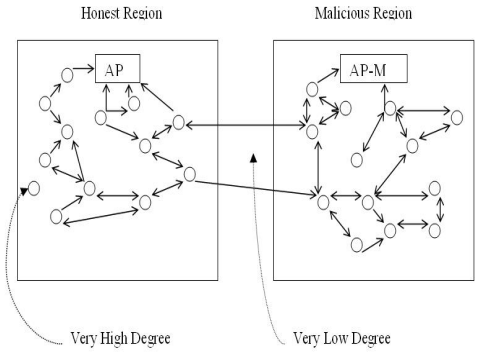


**Figure 7 – Low Degree between Sub-graphs**

novel way to counter this problem based on the attack graph density.

On the system graph describes in Section 3.2, Let $\delta(G)$, $d(G)$ and $\Delta(G)$ represent the minimal, average and maximal degree of the graph. Then the following relation holds true –

$\delta(G) \leq d(G) \leq \Delta(G)$ ----- Equation 4

We define three types of nodes on the graph. An honest node is one that has attributes issues from reputed APs and has a good intention, a malicious node that also has attributes issues by reputed APs but has a bad intention and a Sybil node which does not have attributes issues from reputed APs and has a bad intent.

Honest nodes and malicious nodes exist as fast mixing sub-groups on the graph, each forming a sub-graph G' of high degree but the sub graphs are connected by only a few edges creating a low degree between the sub-groups [10, 12]. This is shown in Figure 7. This is similar to the graph property mentioned in [10]. We call them the honest region and the malicious region on the graph. So within the sub-graphs the degree is very high i.e. closer to $\Delta(G)$ and between the sub-graphs it is very low i.e. closer to $\delta(G)$. This is true because honest nodes interact a lot among themselves and Sybil nodes interact among themselves. For a Sybil node it is more difficult to perform a transaction and receive a confidence score from an honest node. This is true because the very few honest nodes will accept the Sybil node's attributes because they are not certified by reputed APs. On the other hand, the Sybil nodes can arbitrarily provide high confidence scores to each other.

Now, we slightly modify the method of calculating $\Gamma$ as defined in Section 3.3. When the Relying Party finds all the node disjoint paths to the AP, it will calculate $\Gamma$ for the complete graph G (represented as ($\Gamma$, G)) as described earlier. Then the RP removes one of the paths to create a sub-graph H(E',V'). $H \subseteq G$ such that $E' \subseteq E$ and $V' \subseteq V$. Now the RP calculates $\Gamma$ for graph H, represented as ($\Gamma$, H). The RP repeats this by removing each path once and calculating the resulting $\Gamma$. The normalized difference is calculated in a variable $\lambda$ for each removed path as shown in equation 5.

$$\lambda = \frac{(\Gamma, G) - (\Gamma, H)}{(\Gamma, G)}$$ ----- Equation 5

We examine the change in $\lambda$ for each removed path. If the RP and AP both are in the honest region, there will be numerous disjoint paths with similar path confidence values. If we remove one path out of n, the drop in ($\Gamma$, H) compared to ($\Gamma$, G) will be comparable to 1/n. On the other hand, if the RP is in the honest region and the AP is in the malicious region, there will be very few paths from RP to AP. If we remove a path now, the drop in ($\Gamma$, H) compared to ($\Gamma$, G) will be much higher. This can be explained by the fact that M and Sybil nodes will jointly try to increase the confidence value for that path[4] artificially to a very high value. If this path is removed, the drop will be very high. This drop can be characterized and compared against a threshold. We call it the Sybil Threshold. If the value of $\lambda$ is higher than the Sybil Threshold, we say that the confidence value on the path is artificially inflated and remove this path in calculating $\Gamma$. This way we reduce the threat of a malicious user inducing a bogus AP in an attempt to subvert the system.

Using the notation from Section 4.4, the cost of launching a successful self-promotion attack is $\zeta = d.\chi$ *s.t.* $\Gamma >$ *acceptance threshold* and $\lambda <$ *Sybil threshold.*

Unlike the slandering attack where the malicious entities can mount edge attack on source node itself, a self-promotion attack will be more effective if the malicious nodes target well connected nodes with high node weights. Each confidence value from the source node to the AP-M will have two parts, from the source node to the victim[5] node (denoted as $T_{honezt}$) and from the victim node to the AP-M (denoted as $T_{malicious}$). The malicious nodes can make $T_{malicious} \approx 1$.

**Theorem 5** *In the event of a self-promotion attack,* $\Gamma$ *has an upper bound of* $\sum_{i=1}^{\rho} T_{honest,i}$.

***Proof***. For a particular path i, $C_{source->AP-M} = T_{honest,i} * T_{malicious,i}$, since $T_{malicious,i} < 1$, $C_{source->AP-M} < T_{honest,i}$. So for all the available $\rho$ paths, $\Gamma < \sum_{i=1}^{\rho} T_{honest,i}$.

## 5. Discussion

In this section, we discuss how AttributeTrust fulfills some of the model requirements mentioned in Section 3.1.

i) To achieve a short aggregation time, the length of each confidence path is bounded by L. Assuming each entity has n peers, the maximum operations to find all the confidence paths are bounded by $n^L$. After the confidence paths are determined, fetching the node weights and edge confidence values to evaluate the final confidence value should be a trivial task.

ii) The output of the metric is a numeric value that can be directly compared against the acceptance

4. Highly connected malicious node and Sybil nodes will form a single node disjoint path.

threshold to make authorization decisions.

iii) If we evaluate the final confidence value with only a subset of paths, we still get a confidence value. Although this value is lower than the actual value, it is still useful.

iv) This requirement was discussed in detail in Section 4.

In a self-promotion attack, the metric output will degrade if malicious users can provide a confidence path from the RP to the malicious AP. As the number of malicious paths increase, the output degrades proportionally. Thus,
as the attack is intensified, the metric output degrades gracefully rather than the metric breaking down completely at some point.

v) This requirement was discussed in detail in Section 4.1 and 4.3.

## 6. Related Work

**Attribute Aggregation**

If a user has his attributes distributed among multiple providers, these attributes need to be aggregated in a single session and provided to the relying party. Klingenstein [2] looked at this problem in the context of a federation. In his paper, he assumes that the user will authenticate to each IdP and decide which attributes to provide to the SP. In contrast, we propose a policy based approach where the user agent decides which attributes to disclose. Klingenstein argues that this problem is rarely encountered in open systems, while we formulated our AP model especially for open systems.

Chadwick [1] looked at the problem from the point of view of user's privacy and issues in linking the user's different pseudonyms in different attribute authorities (AA). He proposed a protocol for linking the AAs, SP and the user completely under the user's control but the protocol requires intensive participation from the user to authenticate with the AA and linking the AAs to aggregate attributes for each session. AttributeTrust uses a policy based approach where the attributes are aggregated either by the user (automatically through the client program) or the RP. The authentication mechanism for gathering attributes in our framework is also more sophisticated.

Chivers [16] looked at user attribute aggregation as a method of constructing a user's profile via data mining. His method limits the disclosure of attributes to Service Providers by classifying attribute types to protect user privacy. One SP can request attributes from only one type. To classify the attributes, this method assumes that the user has a very good idea of what attributes an SP will require for a particular authorization. In contrast, AttributeTrust uses the minimum information disclosure principle to achieve the same goal without requiring this assumption. Also the user controls the aggregation process at all times, so the user privacy is protected. On the other hand, our model uses an opaque token for delegation which is similar to Chivers recommendation.

**Trust Negotiation**

Winsborough et al. [21] present one of the earlier works on negotiating disclosure of sensitive credentials. They proposed eager and parsimonious strategies and compare their performance.

Squicciarini, et al., [20] presented a framework for negotiating release of sensitive attributes. They discuss several interoperable negotiation strategies for improving privacy. This work is based on the assumption of a standard IdP and we extended these concepts complementing the scope and functionality of the AP. Seamons et al., argued that access control policies protecting sensitive credentials are themselves sensitive and their disclosure should be limited [25].

**Confidence measurements via Reputation systems**

The Eigentrust algorithm was proposed by Kamvar et. al. [13]. Their approach is based on transitive trust. They solve the problem of colluding adversaries by assuming that there are a small number of pre-trusted peers whom the model trusts unconditionally (similar to [15]). We argue that pre-trusted peers may not be available for other applications. Although our work is also based on the notion of transitive trust, it differs from their model because our model does not assume any seed of pre-trusted peers in the system. We solve the problem of colluding adversaries by the appropriate choice of aggregation strategy. Additionally, our work extends previous research by including attribute aggregation and trust negotiation. Hoffman et. al. [17] did a survey of attack and defense techniques for reputation systems. It covers a broad range of attacks and defense techniques. Douceur [11] introduces the Sybil attack, which is a rampant problem in reputation systems. He argues that without a centralized identity authority, it is not possible to control the Sybil attack. AttributeTrust employs a novel method for limiting the impact of Sybil attack by exploiting the property of varying degrees in the system graph. Although we do not eliminate Sybil entities from the system, we effectively isolate them during evaluation of our confidence metric.

Yu, et al., proposed SybilGuard [10] for defending social networks against Sybil attacks. Their model has few 'attack' edges between the honest and Sybil part of the graph. Each edge is a human established trust

5. Victim nodes are victims of successful edge attack by the malicious entity.

relationship among nodes with out of band verification. Our model similarly relies on varying degrees of the system graph and we believe that this property exists in networks where establishing an edge relationship requires a strong credential. However, we argue that out of band verification using phone for each trust relationship assumes high involvement of the user. In AttributeTrust, we use trusted attributes as credentials to create the relationship edge and the human interaction is minimal in the form of providing feedback for successfully completed transactions. Moreover, SybilGuard uses 'random routes' method to provide an upper bound on the number of Sybil nodes as against our method of eliminating the node disjoint paths from confidence calculations. Another difference between the two models is that in SybilGuard, all the edges have equal weights. In AttributeTrust, edge weights (confidence values) and node weights are central to calculating the final confidence values. Another Sybil attack resistant recommendation system was proposed by Seigneur et al., where they implemented their system in email anti-spam settings [6].

Xiong, et al., proposed PeerTrust [14] a reputation based trust model for P2P electronic communities. Their trust model has five factors some of which are fixed and some of which are adaptive in nature. For countering adversary collusion, each node maintains a similarity context with every peer by giving a weight to a peer by seeing how similar their feedback was to all the other common peers. While their model produces good results, this may lead to the problem of 'Groupthink', where new participants will not give a poor feedback to a node which received a good feedback from his peers for fear of reducing its own weight.

Guha, et al., [18] propose a model for propagation of trust in recommendation systems. They propose a model based on transitivity of trust and distrust. Their model predicts trust between any two entities in the system even with few expressed trust relationships per node. However, certain contributions of AttributeTrust like fine grained confidence for each attribute, scaling a user's opinion by his node weight are not in the scope of their work.

Reiter, et al., [5] and Beth, et al., [9] have proposed authentication models based on transitive trust. Their notion of transitive trust is verifiability of target user's public keys which is quite different from ours. Levien, et al., [15] proposed attack resistant trust metrics based on a notion similar to [5] which assumes a seed of trusted peers.

Buchegger, et al., [24] used a Bayesian reputation system model to mitigate slandering attack by eliminating transitive information that deviates substantially from direct information. Our model mitigates the slandering attack by choice of appropriate aggregation strategy.

## 7. Conclusion

AttributeTrust is presented as a framework for attribute aggregation and for evaluating the confidence in these attributes. The model formulation and metric calculation was discussed. It was also shown how AttributeTrust is resistant to common attacks on reputation systems.

## 8. Acknowledgements

This research was supported in part by the National Science Foundation (under Grant CNS-CT-0716252), the state of Georgia, and the Institute for Information Infrastructure Protection. This material is based in part upon work supported by the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001, under the auspices of the Institute for Information Infrastructure Protection (I3P) research program. The I3P is managed by Dartmouth College. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of any of the sponsors.

## 9. References

1. David W Chadwick, "*Authorisation using Attributes from Multiple Authorities*", Proceedings of the 15th IEEE International Workshops on Enabling Technologies (WETICE'06), 2006

2. N. Klingenstein, "*Attribute Aggregation and Federated Identity*", Proceedings of the 2007 International Symposium on Applications and the Internet Workshops (SAINTW'07)

3. Toni Nykänen, "*Attribute Certificates in X.509"*, HUT TML 2000, Tik-110.501 Seminar on Network Security, Helsinki, Finland 2000

4. John Linn, Magnus Nystrom, "*Attribute Certification: An Enabling Technology for Delegation and Role-Based Controls in Distributed Environments*", Proceedings of the fourth ACM workshop on RBAC, pp 121 - 130, 1999

5. Michael K. Reiter, Stuart G. Stubblebine, "*Authentication Metric Analysis and Design*", ACM Transactions on Information and System Security, Vol. 2, No. 2, May 1999, Pages 138–158

6. J.-M. Seigneur, A. Gray, and C. D. Jensen, "*Trust Transfer: Encouraging Self-Recommendations without Sybil Attack*", in Proceedings of the Third International Conference on Trust Management, LNCS, Springer, 2005

7. S.V. Nagaraj, "*Access Control in Distributed Object Systems: Problems With Access Control Lists*," p. 163, IEEE WETICE, 2001

8. Eric Yuan, Jin Tong, "*Attributed Based Access Control (ABAC) for Web Services*", Proceedings OF THE IEEE International Conference on Web Services (ICWS), 2005

9. Thomas Beth, Malte Borcherding and Birgit Klein, "*Valuation of Trust in Open Network*", Proceedings of the European Symposium on Research in Computer Security 1994, U.K.

10. Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, Abraham Flaxman, "*SybilGuard: Defending Against Sybil Attacks via Social Networks*", SIGCOMM'06, September 11–15, 2006, Pisa, Italy

11. John R. Douceur, "*The Sybil Attack*", Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02),7-8 March 2002

12. http://advogato.org/trust-metric.html

13. Sepandar D. Kamvar, Mario T. Schlosser, Hector Garcia Molina, "*The EigenTrust Algorithm for Reputation Management in P2P Networks*", WWW2003, May 20–24, 2003, Hungary

14. Li Xiong, Ling Liu, "*PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities*", IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 7, July 2004

15. Ralph Levien, Alexander Aiken, "*Attack-resistant trust metrics for public key certification*", Proceedings of the 7th USENIX Security Symposium, 1998

16. Howard Chivers, "*Personal Attributes and Privacy*", Department of Computer Science, Univeristy of York, Heslington, York

17. Kevin Hoffman, David Zage, and Cristina Nita-Rotaru, "*A Survey of Attack and Defense Techniques for Reputation Systems*", Tech Report Tech CSD TR #07-013 , Purdue University

18. R. Guha, Ravi Kumar, Prabhakar Raghavan, Andrew Tomkins, "*Propagation of Trust and Distrust*", WWW2004, May 17–22, 2004, USA

19. David Bauer, Douglas M. Blough, and David Cash, "*Minimal Information Disclosure with Efficiently Verifiable Credentials*", to appear in DIM'08 (Fourth ACM Workshop on Digital Identity Management), October 2008, Fairfax, VA, USA

20. A. Squicciarini, E. Bertino, E. Ferrari, F. Paci, B. Thuraisingham, "*PP-Trust-X: A System for Privacy Preserving Trust Negotiations*", ACM Transactions on Systems and Information Security, July 2007

21. William H. Winsborough, Kent E. Seamons, Vicki E. Jones, "*Negotiating Disclosure of Sensitive Credentials*", 2nd Conference on Security in Communication Networks, Italy, 1999

22. Chiba Masayuki, Urushima Kenji, Maeda Yoji, "*Personal Attribute Provider: A Secure Framework for Personal Attribute Exchange on the Internet*", Tran. of IPSJ, Volume 47, No 3, Page 676-685 (2006)

23. Andrew Twigg, Nathan Dimmock, *"Attack-Resistance of Computational Trust Models"*, Proceedings of WETICE 2003

24. Sonja Buchegger, Jean-Yves Le Boudec, "The Effect of Rumor Spreading in Reputation Systems for Mobile Ad-hoc Networks", In Proc. of WiOpt'03, March 2003

25. K. E. Seamons, M. Winslett, and T. Yu, "*Limiting the Disclosure of Access Control Policies During Automated Trust Negotiation*", Network and Distributed System Security Symposium, San Diego, CA, February 2001