Virtual Fingerprinting as a Foundation for Reputation in Open Systems

Adam J. Lee[1] and Marianne Winslett[1]

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL, USA 61801
{adamlee,winslett}@cs.uiuc.edu

Abstract. The lack of available identity information in attribute-based trust management systems complicates the design of the audit and incident response systems, anomaly detection algorithms, collusion detection/prevention mechanisms, and reputation systems taken for granted in traditional distributed systems. In this paper, we show that as two entities in an attribute-based trust management system interact, each learns one of a limited number of *virtual fingerprints* describing their communication partner. We show that these virtual fingerprints can be disclosed to other entities in the open system without divulging any attribute or absolute-identity information, thereby forming an opaque pseudo-identity that can be used as the basis for the above-mentioned types of services. We explore the use of virtual fingerprints as the basis of Xiphos, a system that allows reputation establishment without requiring explicit knowledge of entities' civil identities. We discuss the trade-off between privacy and trust, examine the impacts of several attacks on the Xiphos system, and discuss the performance of Xiphos in a simulated grid computing system.

1 Introduction

Open systems are distributed computing systems in which resources are shared across organizational boundaries. Common examples of open systems include grid computing networks, corporate virtual organizations, disaster response networks, joint military task forces, and peer-to-peer systems. Open systems that attempt to make access control decisions based on the identities of their participants cannot be truly open, because they suffer from scalability limitations as the number of authorized users increases. Recent research has addressed this problem by proposing *attribute-based* trust management systems for use in these environments (e.g., [2, 3, 4, 5, 6, 13, 15, 20, 22]). These types of systems provide an effective and scalable means for making authorization decisions in truly open systems, but depending on their deployment model, may have the side-effect of virtually eliminating absolute identity information.

This lack of absolute identity can be a double-edged sword in that it increases system scalability but also increases user anonymity; this may not be appropriate

in all application domains. In traditional distributed computing, user identity forms the basis of audit and incident response systems, anomaly detection algorithms, collusion detection and prevention mechanisms, and reputation systems. As such, this functionality either does not exist or exists only in extremely limited forms in current attribute-based trust management systems. In this paper, we take a first step towards addressing this problem by describing a method for the linking and correlation of multiple identities used by the same entity in attribute-based trust management systems. We then show how these identities can be turned into *virtual fingerprints* which can be exchanged between entities in the system without leaking sensitive attribute or civil-identity information. Virtual fingerprints act much like fingerprints in the physical world in that they allow multiple actions initiated by an entity to be linked without knowing the civil identity of their owner, thereby forming a solid foundation upon which the types of functionality previously described can be constructed.

To illustrate the promise of virtual fingerprinting, in this paper we show how virtual fingerprints can form the basis of the Xiphos reputation system. Reputation systems will be a necessary part of the open systems of the future, as current research trends are beginning to embrace distributed theorem proving approaches to access control [1, 23]. In these systems, proof fragments and access hints are collected from various parties in the network and used to construct proofs of authorization. Accepting these items from malicious entities could have dire consequences, including unbounded searches for non-existent credentials and the risk of being denied access to a resource which one is actually authorized to access. We show how virtual fingerprinting can be used as the foundation of a reputation system that will allow entities in an open system to gain confidence in information provided by others (including proof hints) without compromising each entity's desire to protect his or her sensitive credentials or identity.

In Section 2, we describe how virtual fingerprints can be derived from the information collected during interactions in attribute-based trust management systems. Section 3 describes the design of a reputation system in which ratings are aggregated by using the virtual fingerprinting mechanism described in Section 2. We also discuss several deployment models for this reputation system, each of which allows for a different balance of privacy and completeness of available information. In Section 4, we discuss the privacy implications of Xiphos, examine the effects of several attacks against the system, summarize the results of a performance study that we have undertaken, and comment on the general applicability of virtual fingerprinting to reputation systems. We then overview related work in Section 5 and present our conclusions and directions for future work in Section 6.

2 Virtual Fingerprinting in Open Systems

Each entity, A, in an attribute-based trust management system has a finite set of credentials, $\mathcal{C}_A = \{c_1, \ldots, c_n\}$, which attest to her various attributes. Although these credentials might never explicitly reference A's civil identity (for

example, they could be X.509 credentials that assert only that their owner has a given attribute), we claim that in practice, $\mathcal{C}_A$ completely describes A. In trust management systems such as PolicyMaker [5], KeyNote [4], QCM [10], Cassandra [2], and various trust negotiation proposals (e.g., [3, 13, 15, 22]), each credential is issued to exactly one owner in order to avoid the group key revocation problem. Thus, if an entity E can prove ownership of some $c \in \mathcal{C}_A$, then necessarily $E = A$.

Since entities may consider some of their credentials to be private, $\mathcal{C}_A$ is in most cases not globally available as a basis of comparison for identity establishment. However, as entities in these systems interact, they collect valuable information about one another even if no civil identity information is explicitly disclosed. Specifically, as entities A and B interact, B learns $\mathcal{D}_A^B \subseteq \mathcal{C}_A$. We will call sets such as $\mathcal{D}_A^B$ *descriptions*.

Definition 1. *A* description *is a subset of the credentials owned by one entity which is learned by another entity in the system. We will use the notation $\mathcal{D}_A^B$ to represent the description of A known by B. It is important to note that for B to accept $\mathcal{D}_A^B$ as a description of A, A must demonstrate proof of ownership of each credential $c \in \mathcal{D}_A^B$ to B.[1] The collection of all such descriptions will be denoted by $\mathbb{D}$.*

Over the course of multiple interactions, B can use previously obtained descriptions to recognize when he is communicating with a familiar entity. For this to be useful, however, the number of useful descriptions which an entity can use must be small. We assert that this is indeed the case; even though an entity can have an infinite number of self-issued or other low-value credentials, only credentials issued by *trusted* third parties will be useful in gaining access to the resources shared in an open system. It should not be possible to obtain an unlimited number of such credentials (e.g., a user should not be able to obtain two drivers licenses), which implies that the set of descriptions which can be learned about an entity will necessarily be finite.

Although descriptions are useful for allowing one entity to recognize another entity with whom she has interacted previously, privacy concerns restrict descriptions from being shared between entities. This follows from the fact that entities may consider some of their attributes to be sensitive: even though B learns some credential c which belongs to A, this does not mean that any arbitrary entity in the system has the right to learn c. To allow certain information contained within a description to be shared between entities, we introduce the notion of *virtual fingerprints*.

Definition 2. *The* virtual fingerprint *associated with a description $\mathcal{D}_A^B = \{c_1, \ldots, c_k\}$ is defined as $\mathcal{F}_A^B = \{h(c_1), \ldots, h(c_k)\}$, where $h(\cdot)$ is a cryptographic hash function. The collection of all such fingerprints will be referred to as $\mathbb{F}$.*

---

[1] The only exception to this rule occurs when c is a delegated credential. In this case, $\mathcal{D}_A^B$ should contain both c and the long-term credential from which c was derived. For obvious reasons, proof of ownership of the long-term credential is not required.

The collision-resistance property of hash functions allows virtual fingerprints to be used as pseudo-identifiers in the same way as descriptions. For instance, if SHA-1 is used to derive virtual fingerprints, we expect that each person on earth would need to hold 2^{47} credentials before a collision would be found, given that the current population is about 6.2 billion $< 2^{33}$ people. Therefore, if two virtual fingerprints overlap, their corresponding descriptions overlap, and thus the two virtual fingerprints both describe the same entity. Since virtual fingerprints mask out the details of a user's credentials, they are more likely candidates for allowing inferred pseudo-identity information to be shared between entities. Note that an entity may have multiple disjoint virtual fingerprints and thus even if two entities have interacted with this entity, they may not be able to agree on this fact based on virtual fingerprints alone. However, the limited number of virtual fingerprints used by an entity, A, in the system (which follows directly from the limited number of descriptions of A) implies that over time, factions of entities who known A by each of her virtual fingerprints will form. Clearly, virtual fingerprints can be used to link and correlate the actions of users in an open system without revealing their private attribute data to entities who do not know it already.

It should be noted that virtual fingerprinting cannot be used in conjunction with all types of trust management systems. For example, virtual fingerprints cannot be derived in systems that use anonymous credentials (e.g., [8, 7]) or hidden credentials [11], since the credentials belonging to an entity are never fully disclosed. In addition, the systems discussed in [8, 7] were designed to prevent actions taken at disparate points in an open system from being linked, and thus prevent any form of distributed auditing. However, there are many types of systems that could benefit from the scalability of attribute-based trust management systems, but require the ability to audit transactions in the system so that users can be held accountable for their actions. Examples include grid computing systems, critical infrastructure management networks, joint military task forces, and disaster management coordination centers. Virtual fingerprinting can pave the way for the adoption of attribute-based trust management systems in these types of high-assurance environments by increasing user accountability and auditability. In the remainder of this paper, we substantiate this claim by describing how virtual fingerprints can form the basis of a reputation system for use in systems such as those described in [2, 3, 4, 5, 6, 13, 15, 20, 22].

3   The Xiphos Reputation System

In this section, we present Xiphos, a reputation system based on the virtual fingerprints described in Section 2. The reputation update equations used by Xiphos are similar to those used in other proposals and could easily be changed as better reputation update mechanisms are proposed; in fact, many of the equations presented in this section are adaptations of those presented by Liu and Issarny in [16] altered to work within our virtual fingerprint collection and analysis framework. Thus, our primary contribution is not the reputation update equations themselves, but rather the framework though which entities can record,

index, and exchange virtual fingerprints obtained during their interactions in a privacy-preserving manner to formulate reputations for entities whose identities may never be fully disclosed.

3.1 Local Information Collection

As entities in an attribute-based trust management system interact, they learn valuable information regarding one another's virtual fingerprints. Formally, as entities interact, they store tuples of the form $T = \langle \mathcal{F} \in \mathbb{F}, r \in \mathcal{R}, \tau \in \mathbb{T} \rangle$, where $\mathcal{F}$ is a virtual fingerprint, r is a rating, and τ is the timestamp of the entity's most recent interaction with the entity described by virtual fingerprint $\mathcal{F}$. We assume that the set of all possible timestamps is $\mathbb{T}$ and that reputation ratings come from some set $\mathcal{R}$ of possible values. To simplify our discussion, in this paper we use $\mathcal{R} = [-1, 1]$. However, in practice it will often be the case that ratings are vector quantities (i.e., $[-1, 1]^n$) that allow an entity to rate several aspects of her interaction with another entity (e.g., both the service quality and recommendation quality). All operations carried out on reputation ratings in this paper can be carried out on vectors, so we use $n = 1$ without loss of generality.

Over time, it is possible that some entity B will learn several non-overlapping virtual fingerprints describing another entity A. Thus, after a tuple $\langle \mathcal{F}_A^B, r, \tau \rangle$ is inserted into B's database, B must condense the set of all overlapping tuples. That is, B will remove the set of all tuples $\mathcal{T} = \{T \mid T.\mathcal{F} \cap \mathcal{F}_A^B \neq \emptyset\}$ from his database and insert a single tuple T' which is defined as follows:

$$T' = \left\langle \bigcup_{T \in \mathcal{T}} T.\mathcal{F}, \frac{\sum_{T \in \mathcal{T}} T.r * \varphi(T.\tau)}{\sum_{T \in \mathcal{T}} \varphi(T.\tau)}, \tau_{now} \right\rangle \tag{1}$$

In the above equation, τ_{now} is the current timestamp and $\varphi(\cdot)$ is a function which computes a factor in the interval $[0, 1]$ which is used to scale the impact of older ratings. One possible definition of $\varphi(\cdot)$ fades ratings linearly over some duration d, though other definitions are certainly possible:

$$\varphi(t) = \begin{cases} 1 - \frac{\tau_{now} - t}{d} & \text{when } \tau_{now} - t > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Equations 1 and 2 form the basis of a local reputation system in which any entity can track her interaction history with any other entity in the absence of concrete identity information; this history can then be used as a predictor of future success. In the following subsections, we describe three ways in which entities can exchange portions of their local histories to form a system-wide reputation system.

3.2 A Centrally Managed Reputation System

Information Collection The simplest types of reputation systems to reason about are systems in which a central server is responsible for storing and aggregating reputation values, such as the eBay feedback system. In a centralized

deployment of Xiphos, the server will store tuples of the form $T = \langle \mathcal{F}_A \in \mathbb{F}, lc \in [0, 1], \mathcal{F}_B \in \mathbb{F}, r \in \mathcal{R}, \tau \in \mathbb{T} \rangle$ where $\mathcal{F}_A$ is a virtual fingerprint of the entity reporting the rating, lc is the server's linkability coefficient for the entity whose virtual fingerprint is $\mathcal{F}_A$, $\mathcal{F}_B$ is the virtual fingerprint of the entity being rated (as observed by the rater), r is the rating, and τ is the timestamp at which this rating was logged. Prior to discussing the calculation of reputation values based on these tuples, we must first explain (1) how the server learns $\mathcal{F}_A$ and (2) the mechanism through which lc is calculated.

For several reasons discussed later in this paper, it is important that the server records one of the rater's virtual fingerprints along with each reputation rating registered in the system. One way for this to occur is for the rater to simply reveal several credentials to the server while reporting his reputation rating. Alternatively, the rater could carry out an *eager trust negotiation* [21] with the reputation server prior to submitting his reputation ratings. An eager trust negotiation begins by one party disclosing his public credentials to the other party. Subsequent rounds of the negotiation involve one party disclosing any credentials whose release policies were satisfied by the credentials that they received during previous rounds of negotiation. This process continues until neither entity can disclose more credentials to the other.

In Xiphos, linkability coefficients are used to weight the reputation rating submitted by a particular entity based on how much the rater is willing to reveal about herself. To this end, the function $\gamma : \mathbb{D} \rightarrow [0, 1]$ is used to establish the linkability coefficient associated with a description (as defined in Section 2) learned about an entity. The exact definition of $\gamma(\cdot)$ will necessarily be domain-specific, but several important properties of $\gamma(\cdot)$ can be easily identified. First, low-value (e.g., self-signed) credentials should not influence the linkability coefficient associated with a description. This prevents an entity from establishing a large number of descriptions that can be used with high confidence. Second, $\gamma(\cdot)$ should be monotonic; that is, an entity should not be penalized for showing more credentials, as doing so increases the ease with which her previous interaction history can be traced. Third, to help prevent ballot-stuffing attacks, the sum of the linkability coefficients derived from any partitioning of a description should not be greater than the linkability coefficient derived from the entire description. More formally, given a description $\mathcal{D} \in \mathbb{D}$, $\forall P = \{p_1 \subseteq \mathcal{D}, \dots, p_k \subseteq \mathcal{D}\}$ such that $\cap_{p \in P} p = \emptyset$, $\gamma(\mathcal{D}) \geq \sum_{p \in P} \gamma(p)$. We discuss and evaluate a particular $\gamma(\cdot)$ function which meets these criteria in the technical report version of this paper [14].

The linkability coefficient is a good metric by which to establish a "first impression" of an entity, as a high linkability coefficient implies that an entity's previous interactions can be more easily tracked. This becomes especially meaningful if the reputation system itself stores vector quantities and can look up a "rating confidence" value for a particular user (such as the *RRep* value stored in [16]). Entities with higher linkability coefficients are more likely to have many meaningful rating confidence scores reported by other entities which could be used to weight their contributions to the system.

Given that the server stores tuples in the above mentioned format, we now discuss how reputation ratings are updated. Assume that after interacting with some entity, the server determines that the tuple $T = \langle \mathcal{F}, lc, \mathcal{F}', r, \tau \rangle$ should be inserted into the database. Prior to inserting this tuple, the database first purges all prior reputation ratings reported by the entity described by $\mathcal{F}$ regarding the entity described by $\mathcal{F}'$. That is, the set of tuples $\mathcal{T}_{old} = \{T \mid (T.\mathcal{F}_A \cap \mathcal{F} \neq \emptyset) \wedge (T.\mathcal{F}_B \cap \mathcal{F}' \neq \emptyset)\}$ are deleted from the database.[2] At this point, T can be inserted. Note that user updates replace older reputation ratings rather than scaling them since users locally time-scale their own ratings according to Equation 1.

Query Processing Having discussed how information is stored at the reputation server, we now describe how queries are processed. If an entity is interested in obtaining the reputation of some other entity whose virtual fingerprint is $\mathcal{F}$, he submits a query of the form $\mathcal{F}_Q \subseteq \mathcal{F}$ to the reputation server. To compute the reputation for the entity with the virtual fingerprint $\mathcal{F}_Q$, the server must first select the set of relevant tuples $\mathcal{T}_Q = \{T \mid T.\mathcal{F}_B \cap \mathcal{F}_Q \neq \emptyset\}$. If any subset $\mathcal{T}_Q^A$ of the tuples in $\mathcal{T}_Q$ have overlapping $\mathcal{F}_A$ components, these tuples will be removed from $\mathcal{T}_Q$ and replaced with a summary tuple of the form:

$$\left\langle \bigcup_{T \in \mathcal{T}_Q^A} T.\mathcal{F}_A, max(\{T.lc \mid T \in \mathcal{T}_Q^A\}), \bigcup_{T \in \mathcal{T}_Q^A} T.\mathcal{F}_B, \frac{\sum_{T \in \mathcal{T}_Q^A} T.r * \varphi(T.\tau)}{\sum_{T \in \mathcal{T}_Q^A} \varphi(T.\tau)}, \tau_{now} \right\rangle$$

(3)

This duplicate elimination prevents the server from overcounting the rating of a single entity A who knows the subject of the query by more than one disjoint virtual fingerprint, each of which overlaps $\mathcal{F}_Q$. Let $\mathcal{T}_Q'$ denote the results of performing this duplicate elimination process on $\mathcal{T}_Q$. Given $\mathcal{T}_Q'$, the reputation associated with the query $\mathcal{F}_Q$ is defined by the following equation:

$$r_Q = \frac{\sum_{T \in \mathcal{T}_Q'} (T.lc * \varphi(T.\tau) * T.r)}{\sum_{T \in \mathcal{T}_Q'} (T.lc * \varphi(T.\tau))}$$

(4)

In short, the reputation returned by the server is the weighted average reputation rating of entities matching the virtual fingerprint $\mathcal{F}_Q$, where each reputation rating is weighted based on both the linkability coefficient of the rater (which acts as an estimator of her rating confidence value) and the age of the reputation rating.

The curious reader might wonder why the set intersection operator is used to define $\mathcal{T}_Q = \{T_i \mid T_i.\mathcal{F}_B \cap \mathcal{F}_Q \neq \emptyset\}$ as the set of matching tuples for a query $\mathcal{F}_Q$ rather than the transitive closure of this operator. While in a network with only honest participants, the transitive closure would give more accurate reputation ratings, it would cause incorrect results to be calculated if cheaters are present

---

[2] Alternatively, these tuples could be saved for historical purposes, but marked as expired.

in the system. As an illustration, consider a system in which some entity E (with virtual fingerprint $\mathcal{F}_E$) is known to have an excellent reputation. A malicious entity M (with virtual fingerprint $\mathcal{F}_M$) could then inflate his reputation by having some third party N (with virtual fingerprint $\mathcal{F}_N$) report a rating for the "entity" whose virtual fingerprint is $\mathcal{F}_E \cup \mathcal{F}_M$, thereby causing the tuple $T = \langle \mathcal{F}_N, lc_N, \mathcal{F}_E \cup \mathcal{F}_M, r, \tau \rangle$ to be inserted into the central database. If the transitive closure of the set intersection operation was then used to define $\mathcal{T}_Q$, any searches for M's reputation would then also include all ratings for E, thereby inflating M's reputation. For this reason, we use only set intersection for query matching, as entities can submit queries derived from virtual fingerprints which they have *verified* to belong to another entity. This further justifies the use of the linkability coefficient as a first impression of another entity, since as the linkability coefficient increases towards 1.0, the information included in $\mathcal{T}_Q$ approaches completeness.

3.3 A Fully Distributed Reputation System

We now describe a fully distributed deployment of Xiphos. In this model, entities calculate reputation ratings for other entities by querying some subset of the other entities in the system and aggregating the results from their local databases. As in the centralized model, queries are of the form $\mathcal{F}_Q \in \mathbb{F}$. Each node queried selects from their local database all tuples which overlap $\mathcal{F}_Q$ (i.e., $\mathcal{T} = \{T \mid T.\mathcal{F} \cap \mathcal{F}_Q \neq \emptyset\}$) and then creates a summary tuple of the form $T = \langle r_Q, \tau \rangle$ to return to the querier. If only a single tuple T' matches the query, then its r and τ components are used to form T, otherwise Equation 1 is used to generate a tuple whose r and τ components are used.

Upon receiving each of these summary tuples, the querier then augments them by adding the linkability coefficient that she has associated with the entity which sent the result. This linkability coefficient can either be cached from a previous interaction, the result of an eager trust negotiation initiated by the querier, or calculated from a set of credentials sent by the other entity along with the summary tuple. Given this collection of augmented summary tuples, $\mathcal{T}_Q$, the querier then computes the reputation rating of the entity whose virtual fingerprint is characterized by $\mathcal{F}_Q$ as follows:

$$r_Q = \omega_{local} * r_Q^{local} + (1 - \omega_{local}) * \frac{\sum_{T \in \mathcal{T}_Q}(T.lc * \varphi(T.\tau) * T.r)}{\sum_{T \in \mathcal{T}_Q}(T.lc * \varphi(T.\tau))} \tag{5}$$

The term $\omega_{local} \in [0, 1]$ represents a weighting factor which allows the querier to determine how much of the reputation rating that she calculates should be based on her previous interactions with the subject of a query (denoted by r_Q^{local}) versus the reputation ratings reported by other entities in the system. In addition to choosing the weight given to the reputations returned by others, users must manually balance the time they spend querying other nodes with the accuracy of the reputation rating that they hope to derive.

3.4 A Reputation System for Super-Peer Network Topologies

The final deployment model which we consider is a reputation system built on top of a super-peer network. Super-peer networks are peer-to-peer networks that leverage the heterogeneity of nodes in the network by using nodes with higher bandwidths and faster processors to act as intelligent routers which form the backbone of the network. In these networks, a small number of so-called "super nodes" act as gateways for a large number of standard peers.

In this model, each super node is assumed to have complete information regarding the virtual fingerprint to reputation bindings stored by each of its client peers; that is, each super node acts as a centralized server as described in Section 3.2. Given a query $\mathcal{F}_Q$, a super node then uses Equations 3 and 4 to compute a local reputation rating, r_Q^S, based on the ratings provided by its client peers. However, in addition to calculating this local reputation rating, the super node can also include the reputations reported by other super nodes. After reissuing the query to each other super node and obtaining $\mathcal{T}_Q$, the set of resulting summary tuples calculated using Equations 3 and 4, the super node computes the aggregate reputation in response to the query $\mathcal{F}_Q$ as follows:

$$r_Q = \omega_S * r_Q^S + (1 - \omega_S) * \frac{\sum_{T \in \mathcal{T}_Q}(T.lc * \varphi(T.\tau) * T.r)}{\sum_{T \in \mathcal{T}_Q}(T.lc * \varphi(T.\tau))} \tag{6}$$

As in the fully distributed model, ω_S is a weighting factor that determines how much the reputation rating calculated from the super node's local peer group is weighted in comparison to the reputation ratings returned by all of the other super nodes.

4 Discussion

In this section, we see that Xiphos is in fact a double-edged sword, and system architects must make explicit choices regarding balancing privacy preservation and completeness of available information when deciding which deployment model to use. We then discuss several attacks on Xiphos, summarize the results of an in-depth analysis of the Xiphos system, and comment on the use of virtual fingerprints in conjunction with other reputation systems.

4.1 Privacy Considerations

Possible Privacy Violations We have identified three types potential privacy violations which may occur as a result of the Xiphos system: leakage of interaction history, discovery of groups of entities with similar attributes, and inference of particular attribute information. Interaction history leaks occur in the centralized and super-peer deployments of the Xiphos system any time that one entity registers a reputation rating for another. This action allows the super peer or central server to infer that the rater and the ratee have interacted in the past. In the fully distributed deployment model, anytime that A answers a

query issued by B, B can infer that A has interacted with the subject of his query. However, leakage of interaction history occurs in every other reputation system that we are aware of, thus we do not discuss it further here.

The second type of privacy violation occurs as a central server or super peer collects large amounts of reputation tuples. Recall that these tuples are of the form $T = \langle \mathcal{F}_A, lc, \mathcal{F}_B, r, \tau \rangle$. After building a substantial database, a malicious server can select all tuples whose $\mathcal{F}_B$ component overlaps a given $\mathcal{F}_Q$ exactly. We now claim that the $\mathcal{F}_A$ components of these matching tuples determine a set of entities in the server's view of the open system who have similar attributes. The justification of this claim comes from the fact that each entity described by some $T_i.\mathcal{F}_A$ was able to determine the same virtual fingerprint for the entity matching $\mathcal{F}_Q$. Thus, each of these entities was able to unlock each of the credentials used to derive $\mathcal{F}_Q$, a feat which requires that each of these entities be able to satisfy the same set of credential release policies. Because these release policies are not always strict conjunctions, we cannot determine that each matching $T_i.\mathcal{F}_A$ has the *same* set of defining attributes, though we can claim that these entities are *similar* in some respects. Note that the similarity of these entities is directly correlated with the restrictiveness of the release policies protecting the credentials used to derive $\mathcal{F}_Q$; more restrictive policies lead to more related entities.

The third type of privacy violation allows certain entities in the system to infer attributes possessed by another entity in the system. In the centralized and super-peer models, this attack is an extension of the previously discussed attack. Consider the case where a server S knows the description $\mathcal{D}_A^S$ of a node A. Let us also assume that some $c \in \mathcal{D}_A^S$ is protected by a release policy, p, which is also known to S (e.g., as a result of a previous interaction). S can then form a query $\mathcal{F}_Q = \{h(c)\}$ and process it using the technique described above, thereby learning the virtual fingerprints of a group of entities who can satisfy p. Since S knows p, he then knows not only that each entity that matched his query is related *somehow*, but also that they satisfy p; that is, S can infer the attributes which cause the similarities between the nodes which match his query.

A Balancing Act To an extent, these attacks can be mitigated by choosing an appropriate deployment model for the Xiphos system. The centralized model makes these attacks easier to carry out, as the server has complete information regarding the reputation tuples registered with the system. By using a super-peer deployment, the information flow is restricted greatly. Both the group discovery and attribute inference attacks are limited to occurring within a single peer group, since super nodes do not have access to each others' databases. Thus, if client nodes restrict their information sharing to super nodes whom they can trust (e.g., super nodes with Better Business Bureau memberships or TRUSTe-issued privacy policies), then they can have some assurance that the super node will not abuse their partial information to carry out these attacks. Limiting the size of peer groups managed by each super node further restricts these attacks. It should also be noted that using the super-peer deployment model does not sacrifice the completeness of information available, as ratings registered by every

peer are still included as the contribution of each super node is folded into the reputation rating calculated using Equation 6. However, unless each super node has a roughly equivalent number of members, ratings may be biased towards the opinions of entities at super nodes with fewer members. Additionally, unless super nodes coordinate to ensure that there is no overlap between their respective peer groups, the accuracy of the reputation ratings calculated using this method may suffer, as malicious peers could register ratings at multiple super nodes.

These attacks can be further limited by using the fully distributed deployment model, as no entity in the system has any sort of complete information. Each entity is restricted to querying a limited number of other entities in the system, as querying each node in turn becomes inefficient as the size of the network grows. Additionally, when issuing the query $\mathcal{F}_Q$, an entity A cannot be sure if the responding entities have matched all of $\mathcal{F}_Q$ or simply some $\mathcal{F}' \subset \mathcal{F}_Q$. This implies that A must carry out the group discovery or attribute inference attacks by issuing queries $\mathcal{F}_Q$ where $|\mathcal{F}_Q| = 1$ to ensure that all matches returned are total matches. Note also, that A will most likely need to know c where $\mathcal{F}_Q = \{h(c)\}$, as otherwise she is simply guessing that $\mathcal{F}_Q$ is an "interesting" virtual fingerprint, which may often be a difficult task. This implies that A is very likely to know p, the release policy for c, as she satisfied p to learn c in the first place. In this respect, the group discovery attack is eliminated, as A is forced to carry out the stronger attribute inference attack. The attribute inference attack is itself no more feasible than trying to determine whether the attribute a attested to by c is possessed by each node in the network directly (e.g., by means of an eager negotiation or another resource access request protocol), thus this attack is no more feasible with Xiphos in place than it would have been without it. This implies that attacks which cause the aforementioned privacy violations can be virtually eliminated by using the fully distributed deployment model, though at the cost of losing the completeness of reputation information.

In addition to choosing an appropriate deployment model, another possible avenue for the prevention of privacy-related attacks involves the use of *obligations*. Obligations are requirements that can be attached to personal information in certain types of trust management systems. For instance, the owner of a digital medical record might attach an obligation to that record requiring that her health care provider send her an email any time this record is shared (e.g., while filing a referral to another physician). In these types of systems, it would be possible for entities to attach obligations to their credentials which limit the ways that other entities can disclose virtual fingerprints including hashes of these credentials. For example, an entity could indicate that any virtual fingerprint including a hash of her Department of Energy security clearance credential may only be released to servers operated by the U.S. government. These types of obligations allow users to reap the benefits of Xiphos while still maintaining some control over their private information. We expect that most entities will allow at least some "interesting" subset of their credential hashes to be included in virtual fingerprints because they will likely interact with other entities who require the ability to obtain their reputation rating prior to interaction.

4.2 Attacks and Defenses

One common attack against reputation systems is whitewashing, in which a user sheds a bad reputation by establishing a new identity. In some cases, this is as simple as reconnecting to the network to obtain a new node identifier, while in others it may involve establishing a new pseudonym (e.g., email address) by which one is known. In Xiphos, nodes are identified by their virtual fingerprints. As discussed in Section 2, users have a limited number of virtual fingerprints, which are uniquely determined by the set of credentials that the user possesses. Obtaining new identities thus reduces to establishing new virtual fingerprints; this requires that a user obtain *all* new credentials, as *any* overlap will link this entity to old ratings. If users are routinely required to use multiple credentials, this process becomes time consuming and involves multiple certificate authorities, thereby making whitewashing an impractical attack for *habitual* cheaters.

In many reputation systems, it is possible for an entity to "stuff the ballot box" by registering multiple ratings for a single entity. Xiphos limits this attack because entities have only a finite number of disjoint virtual fingerprints which can be used to register claims for a given entity. In addition to capping the number of ratings that an entity can register, the virtual fingerprint system also limits the benefits of registering multiple ratings. A properly designed $\gamma(\cdot)$ function will assign lower linkability coefficients to ratings associated with a small rater virtual fingerprint than it will to ratings associated with large rater virtual fingerprints. This means that given a properly designed $\gamma(\cdot)$ function, an entity's influence on the overall rating of another entity will be less if she registers multiple ratings using a large number of small virtual fingerprints than it would have been if she had registered only a single rating using the union of each smaller virtual fingerprint. Such a $\gamma(\cdot)$ function is discussed in [14].

One attack against Xiphos itself involves exploiting $\varphi(\cdot)$. Recall that $\varphi(\cdot)$ is used to weight the contribution of a single tuple to the overall reputation calculated for a query. In centralized or super-peer deployments, entities may try to increase their influence by repeatedly updating their ratings for other entities to keep them current. In the absence of certified transactions and synchronized clocks, there is little that can be done to prevent this problem. However, this attack will have little influence on the ratings calculated by a server if the majority of the users remain honest. Nonetheless, investigating mechanisms for providing certified timestamps is an important area of future work.

One last attack on which we comment occurs when a malicious party M is able to steal some set of credentials $\mathcal{C}'_A \subseteq \mathcal{C}_A$ from another entity A. If M then submits a reputation rating for some entity B described by the virtual fingerprint $\mathcal{F}_B$ while posing as A (by using the stolen credentials $\mathcal{C}'_A$), this rating will overwrite the rating previously stored by A. Though this attack is serious, it is possible in any system in which one entity is able to effectively steal the identity of another. Due to the fact that users in attribute-based trust management systems have many identities (which we have referred to as descriptions in this paper), open systems researchers must focus on making secure identity management easy for users of their systems to prevent these types of attacks.

4.3 Performance Evaluation

As mentioned in Section 3, Xiphos uses reputation update equations similar to those whose convergence behavior was studied in [16]. Rather than studying convergence, we focused on quantifying the effect of using virtual fingerprints as the basis for a reputation system, instead of more traditional identities, in a number of simulated grid computing environments. To this end, we studied the growth rates of local databases over time, evaluated the effect of these more complex local databases on query processing time, and designed and evaluated the impact of a $\gamma(\cdot)$ function for our application domain. The complete results of this study are presented in [14].

We found that when using a conservative tuple eviction policy, the average size of a local reputation database in a network with 10,000 users was approximately 1,500 tuples after 5,000 days of simulated interactions. In a network of 70,000 users, the average local database contained 4,000 tuples after 5,000 simulated days. When executing a prototype Xiphos implementation on a 1.6GHz laptop, Xiphos could process queries on databases of these sizes at throughputs of 600 and 200 queries per second, respectively, without using indexes. The use of a more aggressive, though still reasonable, tuple eviction policy resulted in query throughputs of over 2,200 queries per second on both simulated networks; it is unlikely that the network characteristics of actual grid computing systems would even allow queries to arrive at such a high rate. We also verified that a suitable $\gamma(\cdot)$ function can limit the damages caused by attackers in the system by penalizing them for maliciously using multiple identities. These observations indicate that virtual fingerprints can be used as a reasonable basis for reputation in the open systems of the future.

4.4 Virtual Fingerprinting and Reputation

Virtual fingerprints provide a general notion of pseudo-identity that can be bound to reputation scores in systems where explicit identity information may not be present. Virtual fingerprints are also difficult to change, making whitewashing impractical and reducing the benefits of assuming multiple personalities. In this paper, we described the use of virtual fingerprinting as the basis for one *particular* reputation system. However, the reputation scores bound to virtual fingerprints can be aggregated according to *any* reputation calculation method, provided that the complications arising from the legitimate assumption of multiple identities (in the form of disjoint virtual fingerprints) are addressed.

In particular, systems need to mitigate the effects of malicious users assuming multiple identities to over-influence the system. Additionally, the fact that queries may overlap multiple tuples could lead to problems maintaining precomputed reputation scores at a naive server. In [14], we describe an ontology-based definition of $\gamma(\cdot)$ that prevents malicious entities from over-influencing our simulated grid computing system; similar definitions are likely to be possible in other domains. We also present a method for maintaining precomputed reputation estimates at a centralized server, though space limits prohibit its discussion

here. Similar modifications could be made to other reputation systems (including those not yet developed), thereby enabling them to use virtual fingerprints and extending their applicability to attribute-based trust management systems.

5   Related Work

While the area is too broad to survey in general, papers such as [9, 12, 16, 19] address the design of general-purpose reputation systems. These types of systems assume that entities have established identities in the system and many times suffer from whitewashing and ballot-stuffing attacks. The authors of [17] recommend designing systems which require non-repudiable *evidence* of a transaction be shown in order for a reputation rating to be registered; this certainly prevents an entity from registering multiple claims, but requires that the underlying system support certified transactions. In this paper, we introduced the notion of virtual fingerprints and showed how they can be used as the basis of the Xiphos reputation system. The nature of virtual fingerprints limits the effectiveness of the aforementioned attacks without requiring non-repudiable transaction support from the underlying system.

Other authors have also addressed the privacy versus trust trade-off discussed in Section 4. Anonymous credential schemes (e.g., [8, 7]) assume that privacy is more important than history-based mechanisms such as reputation systems and allow a credential to be used under different pseudonyms; this prevents transactions carried out by a single entity from ever being linked. In [18], the authors discuss this trade-off in detail and show how entities can explicitly reveal linkages between multiple identities to establish trust when needed. Xiphos allows system designers to balance this trade-off by choosing an appropriate deployment strategy. In addition, if Xiphos is used in systems supporting obligations, users can further limit the dissemination of their personal information.

6   Conclusion

In this paper, we presented a method for the linking and correlation of multiple identities in attribute-based trust management systems. We discussed how the descriptions that one entity learns about another can be transformed into opaque virtual fingerprints which can be used as the basis of the Xiphos reputation system. We explored the privacy versus utility trade-off for three deployments of Xiphos and examined the impacts of several attacks against the system. We also highlighted the results of a performance evaluation study of Xiphos, the details of which are available in the technical report version of this paper [14].

One interesting avenue for future work involves the development of index structures for virtual fingerprints to increase the query throughput of the Xiphos system. It is also likely that virtual fingerprinting can be used as the foundation for other useful security services. To this end, we are investigating secure audit, incident response, and collusion-detection systems based on virtual fingerprints.

References

[1] L. Bauer, S. Garriss, and M. K. Reiter. Distributed proving in access-control systems. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, May 2005.

[2] M. Y. Becker and P. Sewell. Cassandra: Distributed access control policies with tunable expressiveness. In *Proceedings of the 5th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY '04)*, pages 159–168, 2004.

[3] E. Bertino, E. Ferrari, and A. C. Squicciarini. Trust-X: A peer-to-peer framework for trust establishment. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):827–842, Jul. 2004.

[4] M. Blaze, J. Feigenbaum, and A. D. Keromytis. KeyNote: Trust management for public-key infrastructures (position paper). *Lecture Notes in Computer Science*, 1550:59–63, 1999.

[5] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *IEEE Conference on Security and Privacy*, May 1996.

[6] P. Bonatti and P. Samarati. Regulating service access and information release on the web. In *7th ACM Conference on Computer and Communications Security*, pages 134–143, 2000.

[7] J. Camenisch and E. V. Herreweghen. Design and implementation of the *idemix* anonymous credential system. In *CCS '02: Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 21–30, 2002.

[8] D. Chaum and J.-H. Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In *CRYPTO '88*, volume 263 of *LNCS*, pages 118–167. Springer-Verlag, 1988.

[9] A. Fernandes, E. Kotsovinos, S. Östring, and B. Dragovic. Pinocchio: Incentives for honest participation in distributed trust management. In *The 2nd International Conference on Trust Management (iTrust 2004)*, pages 63–77, 2004.

[10] C. A. Gunter and T. Jim. Policy-directed certificate retrieval. *Software—Practice and Experience*, 30(15):1609–1640, 2000.

[11] J. Holt, R. Bradshaw, K. E. Seamons, and H. Orman. Hidden credentials. In *2nd ACM Workshop on Privacy in the Electronic Society*, Oct. 2003.

[12] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In *WWW '03: Proceedings of the 12th International Conference on World Wide Web*, pages 640–651, 2003.

[13] H. Koshutanski and F. Massacci. An interactive trust management and negotiation scheme. In *2nd International Workshop on Formal Aspects in Security and Trust (FAST)*, pages 139–152, Aug. 2004.

[14] A. J. Lee and M. Winslett. Virtual fingerprinting as a foundation for reputation in open systems. Technical Report UIUCDCS-R-2006-2691, University of Illinois at Urbana-Champaign, Feb. 2006. Available at `http://dais.cs.uiuc.edu/pubs/adamlee/xiphos_tech.pdf`.

[15] N. Li and J. Mitchell. RT: A role-based trust-management framework. In *Third DARPA Information Survivability Conference and Exposition*, Apr. 2003.

[16] J. Liu and V. Issarny. Enhanced reputation mechanism for mobile ad hoc networks. In *The 2nd International Conference on Trust Management (iTrust 2004)*, pages 48–62, 2004.

[17] P. Obreiter. A case for evidence-aware distributed reputation systems. In *The 2nd International Conference on Trust Management (iTrust 2004)*, pages 33–47, 2004.

[18] J.-M. Seigneur and C. D. Jensen. Trading privacy for trust. In *The 2nd International Conference on Trust Management (iTrust 2004)*, pages 93–107, 2004.

[19] A. A. Selçuk, E. Uzun, and M. R. Pariente. A reputation-based trust management system for P2P networks. In *4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID 2004)*, 2004.

[20] L. Wang, D. Wijesekera, and S. Jajodia. A logic-based framework for attribute based access control. In *2nd ACM Workshop on Formal Methods in Security Engineering (FMSE 2004)*, pages 45–55, Oct. 2004.

[21] W. H. Winsborough, K. E. Seamons, and V. E. Jones. Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition*, Jan. 2000.

[22] M. Winslett, T. Yu, K. E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, and L. Yu. The TrustBuilder architecture for trust negotiation. *IEEE Internet Computing*, 6(6):30–37, Nov./Dec. 2002.

[23] M. Winslett, C. Zhang, and P. A. Bonatti. PeerAccess: A logic for distributed authorization. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS 2005)*, Nov. 2005.