Routing with Confidence: A Model for Trustworthy Communication

APU KAPADIA
Institute for Security Technology Studies (ISTS), Dartmouth College
and
PRASAD NALDURG
Microsoft Research
and
ROY H. CAMPBELL
Department of Computer Science, University of Illinois at Urbana-Champaign

We present a model for trustworthy communication with respect to security and privacy in heterogeneous networks. In general, existing privacy protocols assume independently operated nodes spread over the Internet. Most of the analysis of these protocols has assumed a fraction of colluding nodes picked at random. While these approaches provide promising guarantees of anonymity for such attack models, we argue that trust relationships dominate threats to privacy at smaller scales, and such independence assumptions should not be made. For example, within an organization, all nodes along a chosen path may be physically collocated, making a collusion attack more likely. Users can have varying notions of threat to their privacy – users may not trust nodes located in a particular domain, or consider nodes with low physical security to be a particularly strong threat to their privacy. We present a model for trustworthy communication that addresses users' privacy needs in such environments. Our model also applies to peer-to-peer anonymizing networks such as Tor for finding more trustworthy routes. For example, users may consider nodes operating in a particular country to be untrustworthy. We recognize that users in the network will have different perceived threats and must be allowed to "route around" untrustworthy nodes based on these threats.

Our research makes the following contributions: We present a formalization of trustworthy routing and examine its properties in an effort to understand the boundaries of attribute based trustworthy routing schemes. We propose a model that exposes trust relationships in the network to concerned users. Our policy language allows users to specify qualitative path policies based on their own perceived threat to security and privacy. We define a general quantitative measure of trust that is used to find routes that are most trustworthy based on this measure. We identify feasible and infeasible interpretations of trust by showing how trustworthy routes can be computed efficiently for certain semantic models of trust and by contributing several NP-hardness results for infeasible models of trust.

This research was conducted at the University of Illinois at Urbana-Champaign as part of Apu Kapadia's dissertation research. He was funded by the U.S. Dept. of Energy's High-Performance Computer Science Fellowship through Los Alamos National Laboratory, Lawrence Livermore National Laboratory, and Sandia National Laboratories.

1. INTRODUCTION

We address the trustworthiness of the security and privacy of a user's communication in networked environments. In our study of existing anonymous routing solutions such as Crowds [Reiter and Rubin 1998], Tor [Dingledine et al. 2004], and Mix networks [Chaum 1981], we found that the strong anonymity of these approaches is restricted to widely distributed environments and is problematic in smaller systems. In these protocols, routers are assumed to be independent entities and attackers are assumed to compromise nodes with a certain uniformly applied probability. However, at smaller scales (e.g., an organization) this model does not effectively capture the trust relationships within the network. For example, they fail to provide protection against administrators who may have access to several routers participating in the anonymizing network. A user may distrust nodes running a particular version of an operating system due to a known (unpatched) vulnerability. At such scales, we argue that trust relationships must be exposed to users, who can then make fine-grained decisions about which routers are trustworthy for carrying a user's communication. Furthermore, peer-to-peer anonymizing networks such as Tor can leverage the diversity of nodes within the network to provide stronger trust guarantees by allowing users to discriminate between trustworthy and untrustworthy nodes. For example, a user may not trust Tor nodes located in a particular country or organization. We present a trustworthy communication model for route selection (e.g., in Tor) that allows users to specify richer privacy requirements than the one-dimensional (quantitative) anonymity provided by current solutions. These requirements are based on each individual's perceived threat to privacy, recognizing that users have differing notions of threats to their privacy. Our approach allows users to express their threats effectively. Further, they may use it to find routes within a network that improves the trustworthiness of their private communication. We formalize the notion of trustworthy routing and examine the theoretical limits of such a system. The results provided in this paper have practical significance for any trustworthy routing scheme since it examines feasible and infeasible models of trust within the framework of attribute based route selection.

Our main observation is that routing elements in a heterogeneous network will have several attributes such as domain ID, administrator, physical security, OS version, and attack history. Basing trust on a router's attributes provides users with more flexibility for their individual privacy demands given a set of the user's perceived threats. For example, a user may choose to exclude routers from a particular domain, include only those routers with certified high physical security and exclude intermediate wireless links based on his or her own perceived threat. We also present a quantitative model for trust that allows users to optimize paths based on quantitative metrics of trust, which we call "confidence." For example, a user may assign a high degree of confidence in the DoS resilience of a router, but a low degree of confidence in its resistance to virus attacks. As one example, our model would allow users to find paths with the highest resilience against virus attacks (or in other words, avoiding paths with likely "zombies"). We present a general model for capturing these relationships, and examine the boundaries of feasibility within this model. In Section 5.2 we provide a brief discussion on how confidence can be measured in practice.

We make the following contributions:

—At a higher level, we present a formalization of trustworthy routing that examines the limits of feasible and infeasible trust models for the security and privacy of communication.

—We present a labeled state-transition representation of the various network elements and their attributes.

—We define a language for specifying *qualitative* privacy policies of communication paths based on a user's perceived threat to privacy and show how routes satisfying these policies can be computed efficiently.

—We define a *quantitative* model for measuring the trustworthiness of routes, and efficient algorithms for computing trustworthy paths that satisfy the user's policies. We call this model for trustworthy routing "routing with confidence."

—In addition to our positive results on feasible trust models, we contribute NP-hardness results by showing that several properties of interest in trustworthy routing are computationally hard to satisfy.

1.1 Outline

We provide background and related work on privacy protocols and trustworthy routing in Section 2. In Section 3 we introduce our model for trustworthy communication and describe our language for specifying qualitative path policies in Section 4. In Section 5 we present our quantitative trust model and we provide several results for feasible and infeasible models of trust in Sections 6 and 7. We discuss concrete applications of our model in Section 8 and end with future work and conclusions in Sections 9 and 10

2. BACKGROUND AND RELATED WORK

We begin with a brief overview of privacy protocols. We then provide some background on trustworthy routing.

2.1 Privacy protocols

There has been a substantial amount of research on anonymous communication for the Internet. As a primary goal, these protocols aim to provide sender-anonymity, and some approaches also attempt to provide sender-unobservability through cover traffic. We discuss some of the important work in Internet anonymous routing such as Crowds [Reiter and Rubin 1998], Mix networks [Chaum 1981] and Onion [Reed et al. 1998] routing, and then introduce *Mist* [Al-Muhtadi et al. 2002], which we developed at the University of Illinois specifically for location privacy in ubiquitous computing environments.

2.1.1 *Crowds.* Even though Crowds [Reiter and Rubin 1998] was originally designed for the anonymity of web transactions, the underlying principle is applicable to anonymous routing in general. Each participant is called a *jondo*. The originator of a message picks a jondo from the crowd (possibly itself) and forwards the request to the selected jondo. At each stage, the request is propagated within the crowd with probability p or sent directly to the server (receiver) with probability $1 - p$. The parameter p is assumed to be constant within the crowd, and influences the

average path length between the sender and receiver. Using this simple forwarding scheme, one can show that senders have sender-anonymity beyond suspicion with respect to the receivers. Furthermore, Reiter and Rubin [1998] show that the sender has probable innocence against malicious collaborating jondos for sufficiently large crowd sizes. This is proved using the parameter c, which is the fraction of compromised nodes in the crowd. We argue that this assumption may be reasonable in widely dispersed crowds where routers can be assumed to be independent entities, but does not necessarily hold for crowds in smaller geographic locations where collusion can be highly correlated, e.g., within a building.

2.1.2 *Mixes.* Chaum [1981] proposed an anonymous remailing scheme based on the concept of "mixing." Mail relays in the Mix network would receive emails, and reorder outgoing emails to break the association between incoming and outgoing emails. This was meant to foil traffic analysis by adversaries observing traffic entering and leaving the mail relays. Email messages were encrypted several times to encode the path of remailers. We discuss this technique in more detail in the next section on *onion routing*. The main contribution of Mix networks was to provide a scheme for sender-anonymity that aimed to resist traffic analysis and provide sender-unobservability.

2.1.3 *Onion.* Reed et al. [1998] describe an *onion routing* system meant to establish two-way anonymous communication channels. An "onion packet" contains a message using several layers of encryption. Each router that receives an onion packet can decrypt (or "peel off") the outermost layer of encryption, yielding the identity of the next router in the path chosen by the sender, and an onion packet to be forwarded to the next router containing the rest of the path. The main idea is that each router only knows the previous and next routers of a communication path, providing an anonymous connection between the sender and receiver. Onion routing is an important building block for establishing secure routes of communication since the routers along the path are oblivious to the sending and receiving parties. Dingledine et al. [2004] propose Tor, "the second-generation onion router" and Camenisch and Lysyanskaya [2005] give a formal description of onion routing and provide a provably secure scheme for onion routing. We will assume a secure onion routing scheme in the rest of this paper.

2.1.4 *Mist.* The protocols mentioned above were mainly focused towards providing sender and/or receiver anonymity. It is assumed that the parties communicating want to do so anonymously without revealing their identities, where the location and identity of the person were considered "synonymous." The main contribution of *Mist* [Al-Muhtadi et al. 2002], developed at the University of Illinois, was to recognize and separate the two pieces of identity. In ubiquitous environments, users want to interact with services and other users with their disclosed identities, e.g., Alice and Bob, and are mainly concerned about their location privacy. Hence a system is required whereby users are available (or reachable) for communication in the system while their locations are hidden. How can one provide such a service to its users?

To solve the location privacy problem and allow users to be reachable for communication, *Mist* introduced the concept of *Lighthouses* that served as communication

points for users. This is similar in concept to the NAT endpoints in Tarzan. For example, Alice can register with a Lighthouse through an anonymous channel and make herself available for communication "in the mist." When Bob wants to contact Alice, Bob can communicate with her through her Lighthouse. Like with Onion, each *Mist Router* along the path from Alice to her Lighthouse only knows the previous and next hops. Hence, Alice's Lighthouse can route traffic to Alice without knowing her location. In effect, Alice chooses not to have sender-anonymity, but instead location anonymity (or location privacy). Note that Alice can choose not to disclose her name, in which case Alice will have anonymity in addition to location privacy.

Our work on *Mist* exposed the need for trustworthy routing within an organizational setting. Indeed, a path to a Lighthouse may traverse routers that are not trustworthy with respect to privacy. We now discuss the concept of trustworthy routing.

2.2 Trustworthy routing

We first provide some background on Policy Based Networking (PBN) and then introduce the notion of trustworthy communication within this framework.

2.2.1 *Policy Based Networking.* In PBN, network administrators now have the ability to specify, administer, and enforce an organization's network-access and utilization policies more effectively. PBN has traditionally focused on which users have access to what resources in a network [Sloman and Lupu 2002]. These policies correspond to *mandatory* access control (MAC) and utilization policies that the network, as a system, applies to its users. Our work can be viewed as complementary to such systems since we focus on a user's (discretionary) privacy expectations and preferences governed by the existing mandatory network policies. Users can influence the path of their traffic within this setting. Our motivation stems from the observation that the *discretionary* privacy demands of users have been largely ignored in any formulation of PBN policies, and for communication privacy in general. Our initial work on discretionary path policies within a PBN was published in [Kapadia et al. 2004]. We build on this idea for providing users with a model for trustworthy communication within a network. We now define precisely what we mean by "trust."

2.2.2 *Trust.* "Trust" relates to the degree by which a user believes a certain property to be true. In security, we are interested in security properties such as resilience to attack – "can I trust this system to be virus-free?" or certificate validation – Alice may trust Bob's digital certificate of identity issued by Verisign [ver], but not one issued by some other user Charlie. As defined in [Bishop 2003], an entity is *trustworthy* if there is sufficient credible evidence leading one to believe that the system will meet a set of given requirements. *Trust* is a measure of trustworthiness, relying on the evidence provided. If trust is a numerical measure of trustworthiness, deeming a system as trustworthy may be based on a threshold of some trust value, or simply the system with the highest trust value is deemed to be most trustworthy. We refer to this quantitative notion of trust as "confidence" or "diffidence" as the case may be. In some cases it is useful to measure positive attributes of trust, e.g., reputation based systems. We refer to these positive

measures as "confidence." Likewise, it is also useful to collect negative reports on behavior, e.g., intrusion detection reports. We refer to these values as "diffidence" values. We will refer to the trustworthiness of a route as its "Quality of Protection (QoP)" or more specifically its "path-confidence."

2.2.3 *Trustworthy routing.* Users may be interested in setting up routes that preserve their privacy, in terms of location anonymity or identity anonymity. A user may desire a trustworthy route that is likely to preserve the user's anonymity. Our model enables users to set up routes through routers in a way that does not compromise their privacy. Users can specify trust attributes to avoid certain nodes and prefer some routes over others, rather than relying on the system to make anonymous routing decisions. This system addresses the route-selection phase before using a scheme like Onion routing or Tor. In *Mist*, trustworthy routes can be used for connecting to a Lighthouse.

We now discuss some general approaches towards trustworthy networking. Yi et al. [2001] propose the notion of *secure routing* for ad-hoc military environments. While their work focused on ad-hoc wireless routing environments and the specific credentials of the users and group key management, we present a generalized model based on different types of attributes of users and routers, and trust assumptions between these entities. While Yi et al. [2001] perform route selection using broadcast messages, in our approach communication endpoints are given a graph of the network, and can compute routes based on policies that are not revealed to routers, even when the routers are part of the communication path. Finally, our model also incorporates a quantitative measure of trustworthiness of routes that are complementary to the qualitative routing policies based on attributes. It is also worthwhile to address some of the research in multimedia, which attempts to find paths that satisfy "Quality of Service (QoS)" requirements, much like finding trustworthy routes with high QoP. Routing schemes have been proposed for some discretionary requirements such as bandwidth and latency. Resilient Overlay Networks (RON) [Andersen et al. 2001] have been proposed to discover higher bandwidth (or lower latency) routes on the Internet by attempting to circumvent the standard underlying BGP policies. Selfish Routing [Qiu et al. 2003] examines the effects of non-cooperative routing on the overall performance of the network and proposes algorithms that minimize the cost of selfish routing. Salsano and Veltri [2002] describe a method to incorporate RSVP [Zhang et al. 1993] in Policy Based Networks, where clients can specify QoS demands for their route. Constraint Based Routing (CBR) in Multiprotocol Label Switching (MPLS) [Jamoussi et al. 2002] allows clients to specify certain constraints (again, concentrating on QoS). The network then computes paths for the clients based on these constraints. We address discretionary security requirements of users that desire a higher Quality of *Protection* (QoP) rather than a higher Quality of *Service* (QoS). Moreover, clients do not have the ability to keep their policies private in the QoS protocols, and must disclose them to the network for admission control. We present a model that keeps the user's privacy policies secret from the routers. Lastly, QoS properties such as latency or packet-loss are additive or multiplicative for each node visited (and the penalty is incurred each time a node is visited or revisited). In QoP, the penalty is usually incurred only once for each node visited, making the two measures (and

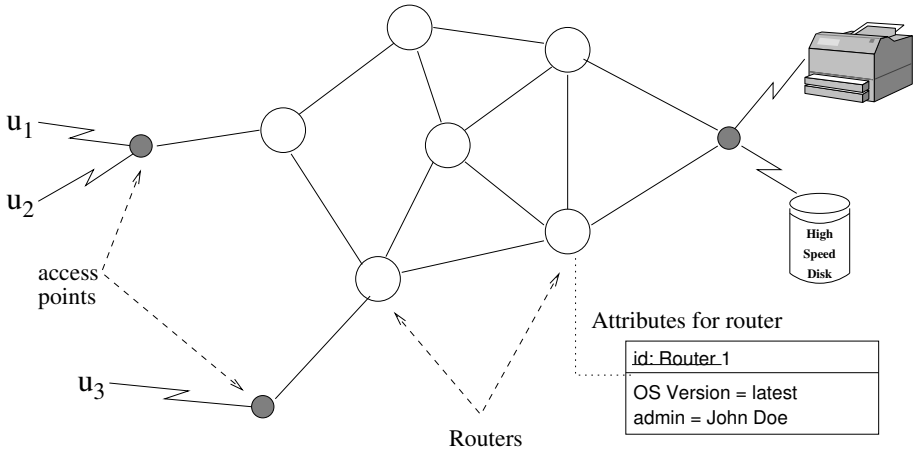


Fig. 1. Architecture Overview

the techniques used to enhance QoP and QoS) incompatible. Consider the property "no node along the path is compromised" where the probability of a single node being compromised is p and is independent of other nodes. For a path of any length, if there are n unique nodes along the path, the probability of no nodes being compromised along the path is $(1-p)^n$, irrespective of the path length. However, the latency of a path does depend on the path length since some latency is incurred every time a node is visited, whether it has been visited before or not. This impacts our choice of policy language presented in Section 4.2.

3. MODEL FOR TRUSTWORTHY COMMUNICATION

Our model consists of three parts: a representation of the network, a policy language for specifying qualitative routing policies, and a quantitative trust model for evaluating the trustworthiness of paths. Using these components, users can specify routing policies based on their perceived threat to privacy and obtain satisfying paths with a high Quality of Protection, or trustworthiness. We begin with a brief overview of the system.

3.1 System architecture

As shown in Figure 1, users connect to our routing infrastructure through *access points*. This could either be a PBN or a network such as Tor. Services can be connected to access points as shown, or certain services may be available at the routing nodes itself (e.g., discovery services that are part of the routing infrastructure). Based on the certified attributes that the user chooses to disclose to the authenticating system (for privacy reasons the user may only disclose a subset of their current attributes), the user is presented with a snapshot of our system consisting of different network elements, including routers, links and servers. Note that this snapshot is a restricted view of the network, reflecting what resources a user

is authorized to use based on the user's disclosed credentials [1], according to the *mandatory* access policies of the organization.

The user hence possesses a logical view of the routers, their attributes, and their connectivity. When a user wishes to communicate with another entity on the network, he or she looks up the access point of the destination and computes a route to that access point. Within this logical view of the network, our framework allows the user to specify qualitative and quantitative policies for trustworthy communication.

In the next few subsections we describe how each part of this process works, along with the trust negotiation and bootstrapping that occurs in the system.

3.2 Assumptions

We assume a network such as a PBN within an organization or a specialized network such as Tor. This allows for the use of specialized protocols for secure and private communication within the organization. Attackers can actively drop, modify, or inject packets into the network and end-to-end encryption can be used to detect such activity. We depend on a suitable Public Key Infrastructure (PKI) and centralized or distributed trust authority for issuing certificates for *attributes*. While this paper is motivated by small scale networks where traditional attack models are insufficient, our model is widely applicable to specialized Internet-wide networks such as Tor where nodes can be issued attribute certificates, and users can select more trustworthy routes based on these attributes.

3.3 Approach

We introduce a model of the network as a labeled state-transition diagram, and use a subset of Constraint Linear Temporal Logic (CLTL) [Demri and D'Souza 2002] based on integer periodicity constraints for discretionary policy specification. These formulas are based on attributes of entities in the network, which may be qualitative or quantitative. This approach allows users to specify qualitative communication path properties based on quantitative attributes, and explore algorithms to discover routes that satisfy users' policies. For example, a user may demand a path that visits only physically secure routers (a qualitative demand) with fewer than 15 intrusion reports (a quantitative demand) in the previous week. In order to capture the effect of dynamically changing trust relationships on the quality of routes our model can discover, we introduce a quantitative measure of trust called *confidence*. For example, attributes of routers maybe be true with a certain degree of confidence. The combination of these approaches allows users to set up routes of high confidence that satisfy their discretionary policies. We show how our policy language can be efficiently interpreted over the network model and be combined with shortest path algorithms for certain models of confidence. As described in Section 2.2.2, confidence or diffidence is a measure of trust for a node. The Quality of Protection (QoP) refers to the trustworthiness of a route, which we will refer to as "path-confidence" to differentiate it from the QoP or "confidence" of an individual node.

Using this metric, we describe different functions to combine meaningfully the confidence values of individual links along a route quantitatively to compute the

[1] We use *credentials* and *attributes* interchangeably since attributes are certified and are presented as credentials

path-confidence or QoP of a path, presenting what we believe is a novel computational model of trust relationships. Confidence values also capture threat by changing the confidence levels in response to exposed threats and vulnerabilities. We show how we can efficiently compute routes that maximize the confidence a user can expect given the current threat model and trust relationships.

Our proposed framework explicitly models static and dynamic trust attributes of both users and network objects to capture the system's evolution. Some attributes of the network elements are dynamic, in the sense that they may change over time. We list different types of attributes network elements and classify them according to whether they are inherent, consensus based, or need to be inferred by the user in some way. We explore the issue of trust management and describe what entities we need to enable certification and validation of dynamic trust attributes.

3.4 Attributes

Since our network model incorporates attributes of network elements, we first define three types of attributes to capture both the static and dynamic nature of evolving trust relationships in our system—*inherent attributes*, *consensus-based attributes* and *inferred attributes*. Routers and links are associated with attribute-value pairs. As we show later, these attributes allow users to specify qualitative path policies and to quantify the trust relationships in the system by associating them with a measure of confidence.

Inherent attributes: These attributes are relatively static characteristics of an entity, which can be certified by a Certificate Authority (CA). Inherent router or link attributes can include physical location, administrative authority, physical security, clearance level, and firewall security. Based on these attributes, users can set up routes that are physically secure and that belong to a certain trusted administrative entity.

Consensus-based attributes: These attributes relate to the behavior of an entity with respect with other entities in the system. For example, routers in the network can vouch for the integrity of neighboring routers if they appear to be routing packets correctly. A compromised router may stop forwarding packets, and neighboring routers would degrade their trust in that router with respect to packet delivery. Users can therefore use these dynamic attributes to set up routes through routers that have been routing packets reliably on a need-to-use basis. This may encourage good behavior of routers within the network. Since these certificates are issued for the current behavior of a router, it is impractical to have the CA issue such certificates.

Therefore, we need a robust and efficient protocol where routers can generate, agree, and distribute these relatively dynamic attributes. Since routers, especially compromised ones, can lie about these attributes, we suggest the use COCA [Zhou et al. 2002], an online certification authority that uses threshold cryptography to issue these certificates. The basic idea is that at least k out of n routers would need to agree on an attribute to issue a certificate for that attribute. COCA comes with built-in intrusion tolerance for Byzantine failures, and is reasonably efficient.

Inferred attributes: While entities in the network may have inherent or consensus-based attributes, users may have reasons not to trust certain routers or links. A user might infer (through probes for example) that certain routers are running

outdated versions of software with a known vulnerability. This is an indicator that the router may be compromised and is not trustworthy. Hence a user may want to avoid such routers. Since these are attributes that the user assigns to routers (or vice versa), these attributes are local to the entity making the inference. No certification is required for such attributes. Other examples include latest patches, daemons running, past behavior observed by the user, etc.

3.5 Routing model

We now present our routing model. As explained before, users can obtain a map of the network that they are authorized to view according to the organizational mandatory policy at startup. This map lists all the routers, and links, and labels each router and link with the set of attributes and attribute-variables that are valid on that router. Users are allowed to update this map with dynamic attributes at any point in time.

We model our network as a labeled state-transition diagram similar to Kripke structures used in model checking [Clarke et al. 2000]. Formally, a Kripke structure is the tuple $M = \langle S, S_0, R, L\rangle$, where S is a set of states, S_0 is the set of initial or start states, $R \subseteq S \times S$ is a transition relation between states, and $L : S \to \mathcal{P}(AP)$ is a labeling function where $\mathcal{P}(AP)$ is the power set of atomic propositions AP. Given a state $s \in S$, $L(s)$ is the set of atomic propositions that are true in s. We augment this model to also represent link attributes.

In the case of attribute-based routing, the set of routers corresponds to the set of states S in the model. If two routers s_1, s_2 are connected then $(s_1, s_2), (s_2, s_1) \in R$ since we assume symmetric links. We overload the definition of L to include the function $L : S \times S \to 2^{AP}$ that maps a link (u, v) to its set of attributes $L(u, v)$. The resulting model is now a labeled state-transition diagram with a labeling function over nodes and edges. Each relation in R corresponds to the connectivity between routers. The set of attributes at each router can be viewed as atomic propositions (or truth valued statements) about attributes in that that state. Therefore the set AP is the set of all possible attribute-value pairs in our system. For example, the attribute-value pair $a = \langle \mathit{OSVersion}, 4.0\rangle$ is an atomic proposition that is *true* for routers with this specific attribute-value pair, i.e., OS Version 4.0. Without loss of generality, we will refer to attribute-value pairs as atomic propositions. In our previous example, the attribute-value pair $\langle \mathit{OSVersion}, 4.0\rangle$ is represented as the atomic proposition a. Note, this set is finite in our model. The set of start states S_0 are specified by the user.

Adding variables: It is clear from the preceding examples that expressing numerical properties can be cumbersome. Consider the attribute-value pairs $\langle \mathit{SecurityLevel}, 5\rangle$ and $\langle \mathit{SecurityLevel}, 4\rangle$. Instead of treating this as two separate attributes, it helps to treat "Security Level" as a variable that takes on different values in different states (or nodes). When these values are real numbers, we allow users to treat attributes as variables and specify arithmetic operations on these variables (e.g., $\mathit{SecurityLevel} < 4$ would correspond to the request "use routers with security level less than 4"). We use the notation $S(s_i)$ to denote the value of variable S in the network node s_i. We discuss the use of arithmetic comparisons of attributes in more detail in Section 5, where we allow comparisons of trust values for properties between routers.

We now summarize our network model:

$$\begin{aligned} AP &: \text{set of atomic propositions} \\ \text{VAR} &: \text{set of variables} \\ M &= \langle S, S_0, R, L, \sigma \rangle \\ S &: \text{set of routers} \\ S_0 &: \text{source router} \\ R &\subseteq S \times S : \text{set of links} \\ L &: S \to \mathcal{P}(AP) \\ L &: R \to \mathcal{P}(AP) \\ \sigma &: S \times \text{VAR} \to \mathbb{R} \\ \sigma &: R \times \text{VAR} \to \mathbb{R} \end{aligned}$$

The labeling function L maps a router (or link) to the set of atomic propositions that are true for that router (or link). The valuation function σ maps a router (or link) and a variable to the value of that variable at the router (or link). For simplicity, we will refer to the valuation of x at a router s_i as $\sigma_i(x)$.

4. QUALITATIVE PATH POLICIES

The user can now define their discretionary policies as path characteristics using temporal logic formulas that can be interpreted over the network model. Different types of temporal logic have been studied extensively in the past [Clarke et al. 2000] to describe properties of infinite computation trees. We focus on finite paths and suitable fragments of temporal logic for this purpose. We believe that the most useful logic for our case is Constraint Linear Temporal Logic (CLTL), which is used specify characteristics of paths based on constraints on attributes. In addition to standard LTL, our proposed fragment of CLTL can express quantitative properties of paths. We do not define the syntax and semantics of LTL as it is well known, but explain how we can use it with quantitative constraints to specify properties of interest. While this approach can be computationally expensive, we show how we can adapt this technique to a computationally inexpensive subset of CLTL and highlight specific characteristics of our problem that make it scalable.

4.1 Tractability of LTL

LTL formulas are a powerful way for users to express qualitative path requirements. Model checkers can be used to generate multiple paths, when they exist, that satisfy these constraints between a source access point and a destination access point in our model. Model checking algorithms for LTL formulas in general have time complexity $O(|M|2^{O(|f|)})$, where $|f|$ is the size of the LTL formula.

In addition to exponential dependence on $|f|$, traditional model checking can also be encumbered by large state spaces (large $|M|$). Some systems with simple high level specifications may result in a "state space explosion." For example, a state transition occurs in a Kripke model when the truth values of the atomic propositions in that state change. As a result, computation trees that represent all possible behaviors of the system by enumerating states and transitions for all

combinations of changes of these values can become very large. Unlike such systems, the state space explosion problem is not a concern for us since $|M|$ is the size of the network. In our case, the attribute certificates are fixed for a particular view of the network. We do not model the changing values of these attributes as different states for each router. Therefore, we can limit the size of our model $|M|$ by number of routers, links, and attributes in our network, and we are only limited by the complexity of algorithms for verifying LTL formulas.

While the overall complexity is low for smaller LTL formulas, finding paths satisfying longer LTL formulas can easily become prohibitively expensive because of the $2^{O(|f|)}$ term. Since we augment this model with quantitative confidence metrics, finding paths of highest confidence that satisfy the LTL formula becomes even more challenging and instead we focus on a fragment of Constraint LTL for which the complexity of finding satisfying paths is *linear* in $|f|$. In addition, CLTL allows users to specify policies using quantitative variables. In effect, our model will reduce to running shortest path algorithms on a directed graph of size $|M|$ after some inexpensive transformations on the graph. We present our policy language in Section 4.2

4.2 Policy language

We now describe the policy language that users can use to specify qualitative constraints on their communication. This is a fragment of LTL with variables, and constraints on these variables. These variables take different values in different states. We refer to the variables as VAR $= \{x_1, x_2, \ldots\}$. VAR takes real values in $\mathbb{R}$. Users specify router and link policies that will be applied independently to each router and link along the path, making it very easy to find paths that satisfy the given policy.

We define the *constraint system* $\mathcal{C} = \{\mathbb{R}, R_1, \ldots, R_n\}$, where $\mathbb{R}$ is the domain of real numbers, each R_i is a relation of arity a_i, such that $R_i : \mathbb{R}^{a_i} \rightarrow \{\mathit{True}, \mathit{False}\}$. An atomic $\mathcal{C}$ constraint over a set of finite variables is of the form $R_i(w_1, \ldots, w_{a_i})$, where each w_i is either a variable or a real-valued constant. Like atomic propositions that represent attributes, these atomic constraints evaluate to *true* or *false*. Mathematical equalities and inequalities are examples of constraints. For example, $x_1 \sim c$ and $x_1 \sim x_2 \oplus c$ where $\sim \in \{<, >, =\}$, $\oplus \in \{+, -, \times, \div\}$, and $c \in \mathbb{R}$, are valid $\mathcal{C}$ constraints.

Let v : VAR $\rightarrow \mathbb{R}$ be a map or valuation of the variables. We also define v to include the identity map over $\mathbb{R}$, in particular $v(r) = r$ for any $r \in \mathbb{R}$. The interpretation of these constraints is as follows.

$v \models R_i(w_1, \ldots, w_{a_i}) \Leftrightarrow R_i(v(w_1), \ldots, v(w_{a_i}))$

In other words, the valuation v satisfies the relation R_i if after replacing all variables with their values as specified by v, the relation evaluates to *true*.

We now define a fragment of CLTL($\mathcal{C}$), i.e., CLTL based on constraint system $\mathcal{C}$. We will call this "policy language" L. We distinguish between two kinds of $\mathcal{C}$ constraints: c_l is a constraint defined with respect to links and c_r is defined for routers. c_l and c_r are relations over variable values at a particular link or router, and additionally, variables in c_l may be prefixed with $\mathbf{P}$ or $\mathbf{X}$ to refer to the values of variables at the incident routers. We will define the semantics of such constraints after presenting the grammar for L:

$$
\begin{aligned}
a &\in AP \\
P &::= \mathbf{G_r}\,\Phi \mid \mathbf{G_l}\,\Psi \mid \mathbf{G_r}\,\Phi \wedge \mathbf{G_l}\,\Psi \\
\Phi &::= c_r \mid a \mid \Phi \vee \Phi \mid \Phi \wedge \Phi \mid \neg\Phi \\
\Psi &::= c_l \mid a \mid \mathbf{P}a \mid \mathbf{X}a \mid \Psi \vee \Psi \mid \Psi \wedge \Psi \mid \neg\Psi
\end{aligned}
$$

We now define the semantics of the policy language. Let $\pi = \langle s_1, \ldots, s_n \rangle$ be a path in the labeled state-transition diagram M. This path represents both routers *and* links. In particular $s_1, s_3, \ldots, s_n$ are routers, and $s_2, s_4, \ldots, s_{n-1}$ are the links $(s_1, s_3), (s_3, s_5), \ldots, (s_{n-2}, s_n)$. Let $\sigma = \sigma_1, \ldots, \sigma_n$ be the sequence of valuations of variables corresponding to routers and links $s_1, \ldots, s_n$ in π. Let S_r be the set of routers in π and S_l be the set of links in π. A discretionary demand by a user includes the policy P and the source and destination pair s, t. The path must be an s, t-path that satisfies P. We define the satisfiability relation for a policy P and path π in labeled state-transition diagram M inductively as follows.

$$
\begin{aligned}
M, \pi &\models \mathbf{G_r}\,\Phi \wedge \mathbf{G_l}\,\Psi \Leftrightarrow M, \pi \models \mathbf{G_r}\,\Phi \wedge M, \pi \models \mathbf{G_l}\,\Psi \\
M, \pi &\models \mathbf{G_r}\,\Phi \Leftrightarrow \forall s_i \in S_r, M, \pi_i \models \Phi \\
M, \pi &\models \mathbf{G_l}\,\Psi \Leftrightarrow \forall s_i \in S_l, M, \pi_i \models \Psi \\
M, \pi_i &\models \neg\Phi \Leftrightarrow M, \pi_i \not\models \Phi \\
M, \pi_i &\models \Phi_1 \vee \Phi_2 \Leftrightarrow M, \pi_i \models \Phi_1 \vee M, \pi_i \models \Phi_2 \\
M, \pi_i &\models \Phi_1 \wedge \Phi_2 \Leftrightarrow M, \pi_i \models \Phi_1 \wedge M, \pi_i \models \Phi_2 \\
M, \pi_i &\models a \Leftrightarrow a \in L(s_i) \\
M, \pi_i &\models \mathbf{P}a \Leftrightarrow a \in L(s_{i-1}) \\
M, \pi_i &\models \mathbf{X}a \Leftrightarrow a \in L(s_{i+1}) \\
M, \pi_i &\models c_r \Leftrightarrow \sigma_i \models c_r \\
M, \pi_i &\models c_l \Leftrightarrow [\ldots, w_j \leftarrow \sigma_i(w_j), \mathbf{P}w_k \leftarrow \sigma_{i-1}(w_k), \mathbf{X}w_l \leftarrow \sigma_{i+1}(w_l), \ldots] \models c_l
\end{aligned}
$$

In other words, policies can specify router policies $\mathbf{G_r}\,\Phi$, link policies $\mathbf{G_l}\,\Psi$, or both. Router policies are boolean combinations of atomic propositions and constraints that must be true at each individual router along a path. Link policies are boolean combinations of atomic propositions, and constraints that must be true (based on the most current knowledge of the attributes) at each individual link along a path. Constraints on links can include variables from the incident routers, allowing the user to specify "one-hop precedence properties." Note that the values of variables and attributes might change after the route has been established. We assume that the users can detect such changes and decide whether they want to compute a new route or not.

We now give some simple examples of our policy language. Let x represent the number of DoS attacks a node has suffered in the past 24 hours. Let y be the width of cable shielding in mm for links in the network. Let z represent the security

level. Let v be the number of virus attacks in the past 24 hours. The policy $\mathbf{G_r}\,(x < 10 \wedge v \leq x) \bigwedge \mathbf{G_l}\,(y > 5 \wedge \mathbf{P}z \leq \mathbf{X}z)$ states that the path must only contain routers with at most 10 DoS attacks and the number of virus attacks don't exceed the DoS attacks, links with at least 5mm shielding, and the order of security levels of routers must be non-decreasing.

4.3 Graph transformation

Given the user's policy we can perform certain transformations on the graph such that any path in the transformed graph will satisfy the user's policy. Furthermore, any path that satisfies the user's policy will be present in the transformed graph. Each atomic constraint (node or precedence constraint) can be evaluated to true or false for a router or link as the case may be. Hence these constraints are atomic constraints that evaluate to *true* or *false* and can be used in boolean expressions with other atomic propositions. Each router of link policy p specified as $\mathbf{G_r}\,p$ or $\mathbf{G_l}\,p$ evaluates to *true* or *false* for the node in question. Routers and links, for which the policies evaluate to false, are removed from the graph. Any path from node s to t in this modified graph will satisfy the overall path policy. In the following section, we will discuss the use of shortest path algorithms to accommodate various quantitative trust models based on the current state of the system.

Our choice of policy language is deliberate, since path automata (such as regular expressions) are not compatible with shortest path algorithms for our model of QoP. Cross-product automata can have several vertices representing the same router in the network. Shortest-path algorithms can accumulate costs for the same router twice by visiting two different vertices corresponding to that router. However, since QoP is computed over *sets* of nodes visited, cross-product automata will yield incorrect solutions. Our policy language makes transformations on the graph, resulting in a subgraph of the network where each router is represented by only one vertex in the modified graph. This can be used with shortest path algorithms for computing QoP, without such problems.

5. QUANTITATIVE TRUST MODEL

Once a user transforms the graph (as described above) of the network satisfying the qualitative attribute requirements, the user would like to set up a route to a destination. One solution would be to obtain the shortest path (in terms of hops) to the destination. However, if the network is under attack, some paths are more trustworthy than others. For example, it may be known that there are intruders in the system with physical access to machines. One would like to degrade trust in routers that have lower physical security. It might be known that certain machines have been compromised without knowing the specific machines. In such a case, users may degrade trust for machines run by certain administrators, or for those machines that are running out of date software. Specifically, certain attributes of routers may be more trustworthy than others. The user may be confident that a router is in a particular domain, but may not be that confident about the router's physical security after a possible break in.

5.1 Confidence of an attribute

We propose a quantitative measure of trust as the *confidence* a user has in an attribute at a router or link. Users can assign confidence levels to attributes of routers. After specifying the qualitative route properties, users can now choose to optimize their routes based on one or more of their attributes of interest. As defined below, each attribute a for a router r (or link (u, v)) is associated with a confidence value $c_r(a)$ ($c_{u,v}(a)$), which can be integer or real valued. We explore the various semantic interpretations of this confidence value and how overall "path-confidence" can be calculated.

Definition 5.1. Given a router $s \in S$ with attributes $L(s)$, a user's **confidence function** $C : S \times AP \rightarrow \mathcal{R}$ returns the **confidence level** for an attribute at a router. We abbreviate the confidence level $C(s, a)$ of attribute a at router s as $c_s(a)$. Similarly we expand the definition to include confidence levels of links. The function $C : S \times S \times AP \rightarrow \mathcal{R}$ returns the **confidence level** for an attribute at a router. We abbreviate the confidence level $C(u, v, a)$ of attribute a at link (u, v) as $c_{u,v}(a)$.

The exact nature of this confidence function will depend on the nature of attributes and how these levels can be composed to compute the confidence or QoP of a path, by combining confidence values of different routers in the path meaningfully. For example, confidence values can represent the probability with which the user believes that the attribute is true or not. It can also represent the number of incidents reported by an intrusion detection system or positive reports submitted by users. In each case these confidence levels must be combined meaningfully to reflect overall path confidence, which we describe in the next section. We discuss how we can compute paths of high overall confidence based on confidence levels of attributes of routers along the path under various semantic models of trust.

5.2 Measurement

Before we continue with our discussion, an important question is how confidence levels can be measured. We based some of our examples on intrusion detection system (IDS) reports. For example, an IDS can record the number of virus or worm intrusions for systems on the network and share these reports with users, possibly attached to the network graph presented to the user. Gossip protocols could be used to share information about neighboring routers, although care must be taken to ensure the integrity of such approaches. Singh et al. [2004] discuss an approach whereby routers can query other routers anonymously to report on their routing table information, and show how the node being queried cannot falsify responses. Various numerical attributes (using attribute variables) can be certified (e.g., security level) and obtained from the router itself, or through a lookup service. When accessing lookup services, it is important to know that queries can leak the policies of users. Hence users must request properties of several (or all) routers to keep their policy contents secret. The use of gossip protocols [Chandra et al. 2001] allows users to passively collect information about other routers, and is a good solution for certified attributes. In this paper we focus on the issue of *using* confidence values by providing a general model that allows the use of several confidence assessment techniques. We aim to characterize what models are feasible and what are not.

5.3 Path-confidence of an attribute

We refer to any simple (no repeated vertices) path from router u to router v as a u, v-path. Similarly a u, v-walk is a path from u to v that may repeat vertices and edges. We assume that u and v are the endpoints of communication.

We now define the path-confidence of an attribute.

Definition 5.2. The **path confidence** $\mathcal{C}_\pi(a)$ for an attribute a along a u, v-path π is obtained by applying a **combiner function** $K(c_1(a), \ldots, c_n(a))$ that takes all the confidence levels $c_i(a)$ of the n nodes s_i along the path π from u to v ($s_1 = u, s_n = v$), and returns a single confidence value for the path in $\mathcal{R}$.

We assume that a combiner function is applied with respect to a single attribute, and omit the "(a)" part in $\mathcal{C}_\pi(a)$ and $c_i(a)$ above for clarity. Users may also want to optimize over the values of variables. For example, let D be the number of DoS attacks a router has suffered within a certain time window. The user may want to find a path that minimizes the sum total of D along a path.

Definition 5.3. The **path confidence** $\mathcal{C}_\pi(D)$ for an attribute variable D along a u, v-path π is obtained by applying a **combiner function** $K(D(s_1), \ldots, D(s_n))$ that takes all the variable values $D(s_i)$ of the n nodes s_i along the path π from u to v ($s_1 = u, s_n = v$), and returns a single confidence value for the path in $\mathcal{R}$.

We assume that if confidence levels are also associated with link attributes, links are subdivided to include a node that represents the link. Hence all confidence levels will be associated with nodes in the graph, which allows us to use shortest path algorithms in a consistent manner. We now focus our attention on how users can optimize path-confidence for a single attribute. In Section 7 we discuss how a user can optimize path-confidence for multiple attributes (multiple combiners).

6. OPTIMIZING A SINGLE ATTRIBUTE

We now explore different combiner functions and how they apply to different models of trust. To illustrate, consider the concept of "weakest link." There may be routers that are highly vulnerable, and it is extremely likely that they will be chosen for attack. The path confidence in this case can be defined as the *minimum* of all confidence values of routers along the path. Here $K(c_1, \ldots, c_n) = \min\{c_1, \ldots, c_n\}$. So when a user needs to pick a path based on its combined confidence value, he or she can avoid paths with low path confidence.

Also consider the following example. A user may conclude that the DoS vulnerability of a router is proportional to the number of incoming links. Hence the user would like a path that minimizes the average sum of incoming links over all routers along a path, but also does not include any nodes with very high connectivity. The user can first eliminate routers with incoming links beyond a certain threshold and then minimize the average. A user may also be interested in minimizing the variance of a certain variable.

First we focus on the multiplicative combiner. A multiplicative measure of path confidence can be used to model various properties of interest to a user: high probability of success of delivery, high probability of no information leakage, high probability that routers along a path will not be compromised, etc.

6.1 Multiplicative combiners

We consider the case when $K(c_1, \ldots, c_n) = c_1 \ldots c_n$, the product of confidence levels of nodes along a path. This multiplicative model of path confidence applies to confidence levels that were computed independently along a path. In this model, a user assigns confidence levels based on the probability of "good things happening" at each node. Assuming independence, the probability of the desired property being true along the entire path is simply the product of all the confidence levels. We now present an efficient method for computing paths of high path confidence under the multiplicative model.

The main idea behind computing paths of high confidence is that by applying the correct weights to edges in a network connectivity graph, we can use shortest path algorithms (that use additive weights) to find paths with highest overall confidence (based on multiplicative weights).

Consider the directed graph G that represents the connectivity of routers specified by the labeled state-transition diagram M. As mentioned earlier, links with confidence levels can be subdivided to include a node that represents that link. For each $s \in S$, we now assign $-\ln(c_s)$ to be the *weight* of all incoming edges to s, i.e., $\{(u, s) \in R : u \in S\}$. We assume non-negative confidence levels in the range $[0, 1]$ (larger ranges can be normalized). We now have a weighted directed graph G'. Consider a source node a and a destination node b.

THEOREM 6.1. *The k-shortest a, b-paths in G' correspond to the k a, b-paths of highest path confidence in G.*

PROOF. *See Appendix Section A.1* □

Since all edge weights are non-negative, Theorem 6.1 allows us to apply k-shortest simple (loopless) path algorithms to find cycle-free paths of highest confidence. For example, Dijkstra's algorithm is the special case when $k = 1$ and will yield a path with maximum path confidence. Several algorithms have been proposed for obtaining the k shortest simple paths in a directed graph. The best known worst case time complexity of these algorithms is $O(kn(m + n\log n))$ [Yen 1971; 1972]. Hershberger et al. [2003] propose an algorithm that provides a $\Theta(n)$ improvement in most cases. For small k (for example, the user may want the 3 highest confidence paths) these algorithms are efficient for all practical purposes. They provide results of their algorithm for large graphs (e.g., 5000 nodes, 12000 edges) based on real GIS (Geographic Information Services) data for road networks in the United States.

In addition to the models we present in this section, we argue that the ability to specify both threat and trust relationships using a combined metric is extremely powerful. We plan to study how these values can vary over time, using sensitivity analysis, stochastic analysis and other techniques. In the next section, we present three example scenarios that showcase the benefits of our new framework.

6.2 Additive combiners

We consider the case when $K(c_1, \ldots, c_n) = c_1 + \ldots + c_n$, the sum of confidence levels of nodes along a path. Finding simple paths of highest path-confidence is NP-hard. For example, the Hamiltonian Path problem can be reduced to finding the highest confidence (or longest path) in a graph with uniform edge-weights, and

then verifying whether it is a Hamiltonian path or not. The same reasoning can be used to show that finding paths of "least confidence" using multiplicative combiners is NP-hard.

Hence we look at the problem where we attempt to minimize the confidence values. For better intuition, we refer to these as diffidence values, and correspondingly refer to path-confidence as path-diffidence. Paths of least diffidence can be solved trivially by using shortest path algorithms. This model can be used in cases where intrusion detection systems may produce negative reports for nodes. Users may want to find paths that minimize the sum of negative reports along the path, corresponding to a path with the least number of known problems. Similarly, a node's incoming degree can be a measure of vulnerability to DoS. A user may want to find a path with the least number of total incoming edges, which could imply a lower amount of vulnerability of the path to DoS.

A user may also want to find a path where the average confidence (or average diffidence) for each node along a path is minimized (respectively maximized). We address the problem of average combiners below, and show that finding solutions for this demand is NP-hard.

6.3 Weakest link

As mentioned earlier, the confidence of the path is the minimum confidence level of nodes in the path, $K(c_1, \ldots, c_n) = \min\{c_1, \ldots, c_n\}$. The path of highest confidence can be computed by sorting the links based on weight. First all links are removed from the graph, and links are added back iteratively in descending order of weight. At each iteration, if a path from s to t exists, then it will be the path of highest confidence. The same can be done for computing paths of least diffidence, where the confidence level is the maximum of confidence level of routers along the path.

For example, consider an attribute that measures DoS resilience. Furthermore the user is certain that there is a DoS attack in the network and would like a path with the highest DoS resilience. Since the DoS resilience of a path is only as good as its weakest link, the user can use this combiner to find a path of highest confidence, or DoS resilience.

6.4 Average combiners

We consider the case when the confidence or diffidence $K(c_1, \ldots, c_n) = \frac{c_1+\ldots+c_n}{n}$, and the user desires a path of least average cost (or least diffidence) or highest average cost (highest confidence).

For example, the user may desire a path that minimizes the average incoming degree for each node. Singh et al. [2004] describe an eclipse attack in overlay networks where malicious nodes are identified by having a higher in-degree. In an eclipse attack, a group of malicious nodes attempts to corrupt routing tables of other nodes in the network, such that all communication in the network is directed through malicious nodes. The authors observe that malicious nodes in this setting would have a high incoming degree and propose an auditing mechanism to ascertain the incoming degree of nodes. In particular, malicious nodes cannot hide their incoming degree because of their proposed anonymous auditing mechanism. A reasonable demand would be to find a path with the lowest minimum average degree, improving the overall confidence in the path with respect to the eclipse

attack. This can be done after eliminating nodes above a certain threshold of incoming degree to avoid the obviously malicious nodes.

We show that these problems are NP-hard by reducing the s,t-Hamiltonian Path problem to finding the minimum or maximum average cost path.

Definition 6.2. **Hamiltonian Path Problem (HP):** Given a directed graph $G = (V, E)$ find an s,t-path that visits all vertices in V. Such a path is called a **Hamiltonian path**.

HP is NP-complete.

Definition 6.3. **s,t-Hamiltonian Path Problem (s,t-HP):** Given a directed graph $G = (V, E)$ and vertices $s, t \in V$, find an s,t-path that visits all vertices in V. Such a path is called an **s,t-Hamiltonian path**

s,t-HP is NP-complete. It is easy to show this by reducing HP to s,t-HP. Given a graph G, construct G' by adding vertices s,t, and the directed edges (s,v) and (v,t) for all vertices $v \in V$. G' has an s,t-Hamiltonian path if and only if G has a Hamiltonian path. Hence s,t-HP is NP-complete.

Definition 6.4. **Minimum Average Cost Simple-Path Problem (MinACSPP)**: Given a graph $G = (V, E)$, with positive vertex weights $w(v)$ for each vertex $v \in V$, and vertices $s, t \in V$, find an s,t-path p that minimizes the average cost of p. The **average cost** of a path p is defined as the the total additive cost of p divided by the number of vertices in p.

Maximum Average Cost Simple-Path Problem (MaxACSPP): Given a graph $G = (V, E)$, with positive vertex weights $w(v)$ for each vertex $v \in V$, and vertices $s, t \in V$, find an s,t-path p that maximizes the average cost of p. The **average cost** of a path p is defined as the the total additive cost of p divided by the number of vertices in p.

THEOREM 6.5. *The Minimum Average Cost Simple-Path Problem (MinACSPP) is NP-hard.*

PROOF. *See Appendix Theorem A.4* □

THEOREM 6.6. *The Maximum Average Cost Simple-Path Problem (MaxACSPP) is NP-hard.*

PROOF. *See Appendix Theorem A.5* □

These results imply that in general it is very hard to compute paths that minimize/maximize path diffidence/confidence. A natural question to ask is whether the shortest average path for various restrictions on hop-length can be computed. In Section 7.3 we show that these related problems are also NP-hard.

6.5 Minimum variance

We consider the case when the diffidence $K(c_1, \ldots, c_n) = \frac{c_1^2+\ldots+c_n^2}{n} - (\frac{c_1+\ldots+c_n}{n})^2$, and the user desires a path of least variance or least diffidence.

For example, the user may want to pick a path with the most consistent (as measured by low variance) set of confidence values within an acceptable range.

We show that the problem of minimizing variance is NP-hard by reducing s,t-HP to finding the minimum variance path.

Definition 6.7. **Minimum Variance Simple-Path Problem (MVSPP)**: Given a graph $G = (V, E)$, with positive vertex weights $w(v)$ for each vertex $v \in V$, and vertices $s, t \in V$ such that $(s, t) \notin E$, find an s, t-path p that minimizes the variance of weights for the set of vertices in p.

We assume that s, t are not directly connected by a single edge, because then the solution is trivial. The path $\langle s, t \rangle$ has variance 0 and is the minimum variance path.

THEOREM 6.8. *The Minimum Variance Simple-Path Problem (MVSPP) is NP-hard.*

PROOF. *See Appendix Theorem A.6* □

6.6 Approximation techniques

For minimum/maximum average and minimum variance weight s, t-paths, picking a large value for δ such as n^2, the approximate solutions to these problems will closely approximate solutions to the longest s, t-path problem. Karger et al. [1993] show that unless $\mathbf{P} = \mathbf{NP}$, there is no polynomial time algorithm that can find a path of length $n - n^\epsilon$ for any $\epsilon > 1$. This indicates that there is very little hope to approximate these problems.

7. OPTIMIZING MULTIPLE ATTRIBUTES

Consider the case where a user may want to maximize confidence along a path for two or more attributes. In our model this is the same as applying two or more combiners to the labeled state-transition diagram.

There has been considerable work in the QoS community that addresses finding network paths that minimize multiple constraints. It can be shown that minimizing two additive or multiplicative (or a combination of the two) constraints is NP-hard. Specifically, given n attributes with the additive combiner, and thresholds $L_1, \ldots, L_n$ for each attribute, finding a path with costs $m_1, \ldots, m_n$ for each attribute such that $m_1 \leq L_1, \ldots, m_n \leq L_n$ is NP-complete [Wang and Crowcroft 1996]. This immediately implies hardness results for multiple attributes in the additive or multiplicative models.

7.1 Unifying multiple attributes

We present an approach that unifies confidence for each node, after which a single combiner can be applied. Specifically, multiple attributes are treated as a single attribute with a unified confidence value, after which our results for single attributes can be applied.

7.1.1 *Tractable additive models.* Consider the two attributes "DoS attacks" and "Worm attacks." Each attribute has diffidence equal to the number of intrusion detection reports for that attack. A user may want to pick a path that minimizes the number of "DoS and Worm attacks." In this case the single attribute "DoS and Worm attacks" can be unified by adding their diffidence values. The user can also choose to weight each attribute. For example, the user may consider DoS attacks more important, and unify the DoS Attack and Worm Attack diffidence values d and w as $0.8d + 0.2w$. In fact this is semantically the same as considering

each combiner separately, and minimizing a linear combination of each additive combiner.

Formally, consider the n attribute variables $a_1, \ldots, a_n$. For a given path π, their individual path confidences are represented as $\mathcal{C}_\pi(a_1), \ldots, \mathcal{C}_\pi(a_n)$. As mentioned earlier, finding a path π such that $\mathcal{C}_\pi(a_1) \leq L_1, \ldots, \mathcal{C}_\pi(a_n) \leq L_n$ for supplied thresholds $L_1, \ldots, L_n$ is NP-complete. However, finding a path that minimizes $w_1\mathcal{C}_\pi(a_1) + \ldots + w_n\mathcal{C}_\pi(a_n)$ is equivalent to minimizing the additive cost $\mathcal{C}_\pi(a)$ of unified attribute a, where for each node k, $c_k(a) = w_1c_1 + \ldots + w_nc_n$. For attribute variables, we would have $a(s_k) = a_1(s_k) + \ldots + a_n(s_k)$. This is easy to prove as in [Jaffe 1984].

This model would apply more easily to attributes with low correlation. In general it is reasonable to assume that DoS and Worm attacks are unrelated. For example, the DoS vulnerability of the node is a function of its connectivity, whereas vulnerability to worms is related to the current version of software. While unifying attributes with a high degree of correlation can be done by more complicated unifiers, we use the linear combination model for its tractability.

7.1.2 *Tractable multiplicative models.* We focus on the model where confidence levels are equal to the probability that the attribute is true. We assume that these probabilities are independent. In this case, multiple attributes at a router can be unified into a single attribute using boolean connectives. It is easy to compute the overall probability for expressions such as $a_1 \wedge \ldots \wedge a_n$ or $a_1 \vee \ldots \vee a_n$ using standard combinatorial rules. We assume unifiers of this form. For example, unified attribute a defined as $a_1 \wedge a_2$ will have the confidence $c(a_1)c(a_2)$, while a defined as $a_1 \vee a_2$ will have the confidence $c(a_1) + c(a_2) - c(a_1)c(a_2)$. This allows us to meaningfully unify attributes under the independent probability model. Unified attributes now have a single probability (or confidence) at individual nodes, and the multiplicative combiner can be use to find paths of highest confidence.

For a more complicated combination of boolean connectives we assume a user supplied formula for calculating the overall probability or the use of available tools. Composing events as arbitrary boolean expressions is common in Fault Tree Analysis (FTA) [FTA 1981] and several FTA tools such as Galileo [Coppit and Sullivan 2000] and SAPHIRE [sap] are available for computing the overall probability of the defined event.

We now address more complicated privacy demands where a user may want to visit k or more distinct nodes to make a traceback attack harder to execute by an attacker. In other words, an attacker trying to expose the location of the user will have to retrace the path through these routers, and a user would like to select such a path with high confidence.

7.2 Visit k distinct nodes

In QoS, one might be interested in bounding the number of hops by k. However, in security applications, users might want paths that visit *at least* k nodes to thwart a trace-back attack. We examine the viability of such a demand.

Consider an attribute variable such as node ID d. We consider the case when $K(d(s_1), \ldots, d(s_n)) = 1$ if $n = k$ and 0 otherwise. Finding an s,t-path of highest non-zero confidence is equivalent to finding a simple path from s to t that visits

exactly k or at least k nodes (s,t-k-path or s,t-k^+-path). Such a property would be of interest to thwart traceback attacks to expose a user's location, but this problem is NP-complete [Alon et al. 1995]. Indeed if we set $k = |V|$, then a solution to this problem will yield an s,t-Hamiltonian path for any graph $G = (V, E)$ if and only if one exists. Hence the minimum weight simple k-path and k^+-path problems are NP-hard optimization problems. We call these problems k-MWSP and k^+-MWSP. Note that the problem of finding an s,t-path with at most k nodes (s,t-k^--path) is easily solved by finding the shortest path from s to t and testing whether the length is at most k, however we are interested in k as a *lower bound* for security against traceback attacks.

If we relax the restriction on simple paths to allow walks (vertices and edges can be repeated), the problem of finding an s,t-k-walk is trivially solvable for undirected graphs. Since the graph is undirected, a walk can be constructed that will visit k distinct nodes by first finding the shortest path from s to t. If there are more than k nodes in this path, the desired walk does not exist. If there are less than k nodes in this walk, then neighbors to this walk can be successively inserted to increase the distinct nodes visited by 1 with each iteration. More precisely, at the beginning of each iteration, there will exist at least one vertex v in the walk with a neighbor w in the list of unvisited vertices. This is guaranteed by the fact that the graph is connected. Replace v in the walk with v, w, v.

We now present a high level algorithm for finding an s,t-walk in any directed graph that visits at least k vertices. For a directed graph G, this algorithm finds an s,t-walk that visits the most possible distinct vertices, and hence will trivially satisfy the "at least k" requirement. If the path returned by this algorithm visits fewer than k distinct vertices, then we know that no such path exists.

Decompose graph into Strongly Connected Components (SCC)
//Linear time decomposition $O(|V| + |E|)$ [Tarjan 1972]
for each SCC S **do**
  assign weight $-|S|$ to each incoming edge to S
  //where $|S|$ is the number of vertices in S
end for
let S_s, S_t be the components containing s, t respectively
find the minimum cost path from S_s to S_t
//The SCC graph is a DAG, and minimum cost algorithms for graphs with negative weights can be applied

This algorithm yields a path p from S_s to S_t in the SCC graph that maximizes the sum of vertices of each SCC (or cost $-c$). Starting from s, a walk can be constructed that visits all nodes in each SCC of p, and ending at t. This will be an s,t-walk that visits the maximum number of distinct vertices (c vertices). If $c \geq k$ we can use this walk as an s,t-k^+-walk. If $c < k$, no such walk exists.

For directed graphs in general, we are not aware of the complexity of finding a walk that visits exactly k distinct vertices. However, we show that the minimum cost version of finding a walk that visits k distinct vertices, and optimizes another confidence metric is NP-hard. Hence in general, it is hard to find optimal walks or paths in networks with a specified number of distinct nodes even if we allow repetition of nodes.

Definition 7.1. **k-Distinct Vertex Minimum Weight Walk Problem (k-MWWP):** Given a graph $G = (V, E)$, with positive vertex weights $w(v)$ for each vertex $v \in V$, and vertices $s, t \in V$, find an s,t-walk p that visits $k < |V|$ distinct vertices (we will call this an s,t-k-walk), and minimizes the weight of walk p. The weight $w(p)$ of walk p is defined as the the total additive cost of the *set* of vertices in p. $w(p) = \sum_{v \in p} w(v)$. Hence the cost of visiting a vertex is incurred only once.

THEOREM 7.2. *k-Distinct Vertex Minimum Weight Walk Problem (k-MWWP) is NP-hard.*

PROOF. *See Appendix Section A.3* □

Since finding minimum weight s,t-walks of length k is NP-hard, an interesting question is whether one can find minimum weight s,t-walks with *at least* k distinct vertices. Earlier we showed that finding s,t-k^+-walks without a cost metric can be done in polynomial time. However we show that the minimum cost version of finding s,t-k^+-walks (we call this problem k^+-MWWP) is NP-hard.

THEOREM 7.3. *k^+-Distinct Vertex Minimum Weight Walk Problem (k^+-MWWP) is NP-hard.*

PROOF. *See Appendix Section A.4* □

7.2.1 *Approximation techniques.* While we have shown that finding node-weighted minimum weight s,t-walks in a network that visit k or at least k distinct nodes is NP-hard in general, approximation algorithms may yield acceptable solutions. We leave this to future work, but mention relevant research here. For the simpler node-weighted minimum weight k-cardinality tree problem on graphs with undirected edges Mladenović and Urošević [2004] present a heuristic based on variable neighborhood search and provide performance results under various scenarios. Blum and Ehrgott [2003] show that problem can be solved in polynomial time if the graph contains "exactly one trough." They also present several local search heuristics for the problem, and provide an extensive discussion on related solutions for this problem. Hence, it may be useful to compute approximate solutions based on these heuristics to find paths of acceptable confidence, if not the highest confidence.

7.3 Scoped minimum average cost

We now return to the problem of finding the minimum average cost path in a graph and explore the complexity of s,t-k-walks of minimum average cost.

Definition 7.4. k **Minimum Average Cost Simple-Path Problem (k-MinACSPP):** Given a graph $G = (V, E)$, with positive vertex weights $w(v)$ for each vertex $v \in V$, and vertices $s, t \in V$, find an s,t-k-path p that minimizes the average cost of p. The **average cost** of a path p is defined as the the total additive cost of p divided by the number of vertices in p.

k^+ **Minimum Average Cost Simple-Path Problem (k^+-MinACSPP):** Given a graph $G = (V, E)$, with positive vertex weights $w(v)$ for each vertex $v \in V$, and vertices $s, t \in V$, find an s,t-k^+-path p that minimizes the average cost of p. The **average cost** of a path p is defined as the the total additive cost of p divided by the number of vertices in p.

k^- Minimum Average Cost Simple-Path Problem (k^--MinACSPP): Given a graph $G = (V, E)$, with positive vertex weights $w(v)$ for each vertex $v \in V$, and vertices $s, t \in V$, find an s,t-k^--path p that minimizes the average cost of p. The **average cost** of a path p is defined as the the total additive cost of p divided by the number of vertices in p.

k Minimum Average Cost Walk Problem (k-MACWP): Given a graph $G = (V, E)$, with positive vertex weights $w(v)$ for each vertex $v \in V$, and vertices $s, t \in V$, find an s,t-k-walk p that minimizes the average cost of p. The **average cost** of a path p is defined as the the total additive cost of the set of vertices in p divided by the cardinality of the set of vertices in p.

We prove that these problems are NP-hard optimization problems in Appendix Section A.5, along with a list of five related problems that are also NP-hard.

7.4 Dealing with hardness

In Sections 6.6 and 7.2.1, we present related work in approximation. In particular, it is not expected that viable solutions exist for minimum/maximum average cost and minimum variance. Using our policy language users can qualitatively eliminate nodes with high values diffidence or low values of confidence. This can help find paths with low overall average diffidence or high average confidence. For minimum variance, a user may specify that any link for which the confidence values at its incident routers differ by more than δ is to be avoided. Hence every hop will avoid a large change in confidence, thus providing lower variance. The user can translate their problem into acceptable one-hop precedence properties as an alternative to the more general minimum variable problem. This approach will efficiently yield paths that satisfy the user's modified policy.

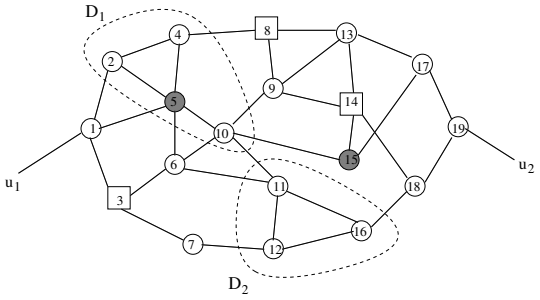
8. APPLICATIONS

We present three concrete examples that use our framework. We present the first example in more detail, and suggest two other uses.

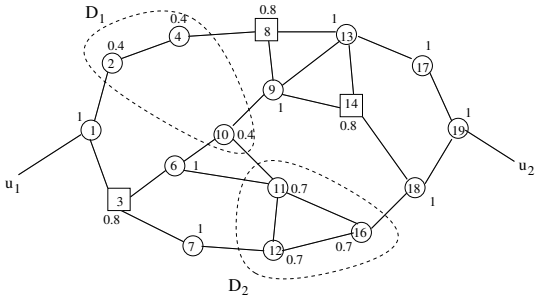
8.1 High-performance and military environments

Consider an MLS (Multilevel Security system) user u_1 with sensitivity level $Confidential$ in compartment $\{Navy\}$, connected at the access point s_1. User u_2 has security clearance $\{Confidential, \{Army\}\}$ and is connected at access point s_{19}. Based on u_1's clearance (u_1 chooses to only reveal this, not its identity), the system presents the user with a logical view of the network as shown in Figure 2(a). All routers in this system are cleared for $\{Confidential, \{Army, Navy\}\}$. For simplicity we look at only two inherent attributes: *physical security* and *Domain*. The attribute *physical security* can take any value in $\{1, 2, 3, 4\}$. Unshaded nodes represent routers with physical security values 3 or 4, shaded nodes represent values 1 or 2. The attribute *Domain* can take any value in $\{1, 2\}$ (D_1 and D_2 in Figure 2(a) correspond to domains 1 and 2). D_1 can correspond to confidential network owned by the Army for example.

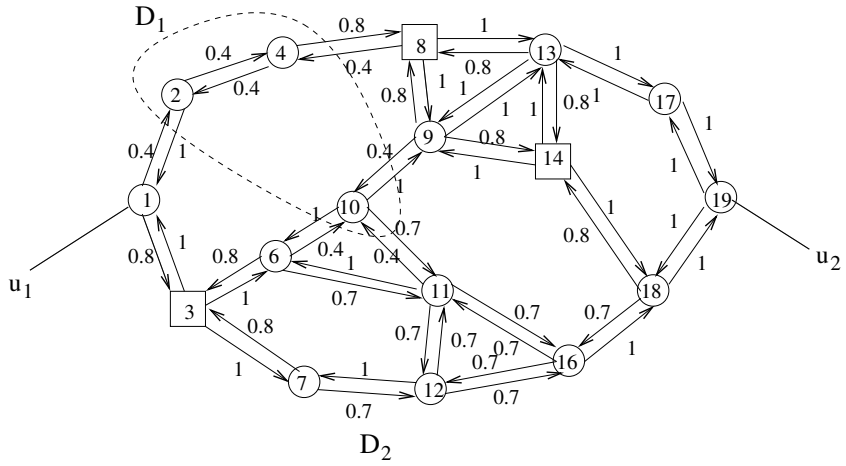
User u_1 desires to communicate with u_2 and determines that u_2's access point is s_{19}. We assume a network component analogous to dynamic DNS which can respond with a user's current access point (u_2 has chosen to register its access



(a) Logical view for u_1



(b) Resultant view based on user's policy, including confidence levels



(c) Resultant digraph for use with k-shortest path algorithms

Fig. 2. Military network example

point with the service). Now u_1 has been informed by trusted sources that there is an intruder physically located on the premises, and that low physical security routers should be excluded. u_1 specifies the following constraint formula $\mathbf{G_r}$ *physical security = high*. This eliminates s_5, s_{15} from the logical view and results in the graph shown in Figure 2(b).

The user now wishes to optimize paths based on a unified attribute "x" based

on *OS version, Domain*, and *physical security*. Under the independence assumption, confidence values of these attributes are unified into one confidence value by multiplying their individual confidence levels. In each case, the confidence value represents the probability with which the user believes the node is not compromised. First, all the confidence values for each attribute at all the nodes are set to 1.

By means of network probes, the u_1 determines the inferred attribute *OS version*, which can be *outdated* (square nodes) or *latest* (round nodes). Routers with outdated operating system versions (*OS version = outdated*) have their confidence levels multiplied by 0.8 since they may be compromised. Lastly, u_1 would like to avoid machines in domain D_1 because of a suspected insider attack in that domain. u_1 multiplies the confidence levels of routers in this domain by 0.4. u_1 has experienced large delays when routing through D_2. Suspecting worm activity, u_1 degrades confidence in those routers by multiplying their confidence levels by 0.7. Figure 2(b) shows these confidence levels for each node. Figure 2(c) shows the resulting digraph with multiplicative weights. As described in Section 5.3 we replace these weight by their negative natural logarithm, and then apply k-shortest path algorithms [Hershberger et al. 2003] to obtain the three paths of highest confidence. In this example it is easy to see that the following are paths with the three highest path confidences: $\langle 1, 3, 7, 12, 16, 18, 19\rangle$, $\langle 1, 3, 6, 11, 16, 18, 19\rangle$, and $\langle 1, 3, 6, 10, 9, 13, 17, 19\rangle$. The first two paths have a path confidence of 0.392 (with respect to the logarithmic weights, the total weight is 0.9365), and the third has a path confidence of 0.32 (weight 1.139).

Once these three paths are obtained, the user needs to set up a path through the routers. This is done using a scheme that encrypts the packet multiple times, based on the routers in the path, as in *onion* routing [Reed et al. 1998], since public keys of routers are assumed to be well known to u_1. The user encrypts the path in reverse order using the keys of the routers in the reverse path. Each subsequent router decrypts the received route setup packet to obtain the next hop and an encrypted route setup packet for the next router. This technique hides the path from the routers, which only know the previous and next hops in the path. By means of this route setup, u_1 can establish the chosen path to u_2. Packets from u_2 to u_1 are simply forwarded on the reverse path.

8.2 Ubiquitous computing

The previous example provided a detailed overview on how our system works in a military environment. In this section we briefly discuss applications to ubiquitous computing. Users in ubiquitous computing environments seamlessly interact with numerous devices and services. In such an environment *discovery* of services, and access to such services is one of the main applications. However, with such an environment it is very easy for the ubiquitous system to track a user's movements or record user patterns. Using our system as a basic infrastructure or service, users can maintain their privacy. Users only need to reveal as much information as necessary to get a logical view of the ubiquitous environment. This can be achieved by trust negotiation as described in [Yu et al. 2003]. In a university setting, a user may want to avoid using routers or services that belong to other research groups, and eliminate these by using global property specifications. While connecting to

certain services, a user may choose to maintain location anonymity (for example, using *Mist* [Al-Muhtadi et al. 2002]) by creating a route to a Lighthouse that is hard to trace back.

8.3 Peer-to-peer overlay networks

We consider peer to peer networks where it is feasible for users to obtain topological information of the overlay. We assume the user can form a logical view of the overlay network based on information available to the user. We have presented a general model for use with arbitrary topologies. Currently, networks such as Tor are fully connected networks, for which several trust models that are otherwise NP-hard can be computed efficiently. For example, finding the shortest path that visits at least k nodes can be computed by simply selecting the k least cost nodes in the graph. However, we envision structured peer-to-peer networks that restrict links between different nodes. For example, a particular Tor router may only accept connections from a subset of trusted Tor routers. As diversity is exploited in such networks, we believe that the topologies will also be more varied than a fully connected topology.

9. FUTURE WORK

In this paper, we discussed several semantic models of trust. In Section 6.1 we showed how multiplicative combiners can be used for probabilistic models of trust. For example, the confidence value of 0.9 for the attribute "Physically Secure" can mean that the probability that the machine is physically secure is 90%. We showed how the overall confidence of the path, the probability that the entire path is physically secure, is simply the product of these probabilities, assuming they are independent. In modeling, one attempts to abstract or approximate the real probability distribution with one that is tractable. For example, Markov models in queueing theory assume a memoryless distribution for arrivals, which greatly simplifies the mathematics. While it is certainly true that security failures in a network are not independent, we would like to examine the space of probability models that are more powerful than the independent probability model, but still computationally efficient to compute. In particular, we would like to avoid approaches with time or space complexity that is exponential in the size of the network. We have been exploring the use of Markov Random Fields and Bayesian networks to represent correlated confidence values. While these models appear to be intractable in general, we would like to identify restrictions on these models that yield tractable solutions. For example, Sahner and Trivedi [1986] explore combinatorial approaches in conjunction with Markov models to constrain the state-space. Such approaches may yield useful and tractable models for trustworthy routing. We would also like to study more powerful policy specification languages for trustworthy routing. Ultimately, user studies are required to identify a usable policy language or a higher level policy specification tool, which would allow regular users to specify privacy policies in a more intuitive way. We also identified several semantic models of trust for which it is NP-hard to find paths of highest confidence. Further research is required for approximation techniques and heuristics for computing paths with reasonably high confidence even if the optimal solutions are elusive.

10. CONCLUSIONS

In this paper, we argued that users must be given sufficient control over their communication in an organizational network, such as a ubiquitous computing environment. We presented a model that allows users to specify discretionary privacy properties of routes based on the attributes of routers and links in the network based on perceived threat. These attributes can represent boolean as well as real-valued properties. Our policy language allows users to specify global and one-hop precedence properties based on attributes of links and routers. We discussed various representations of trust and showed how trustworthy routes can be computed efficiently for various semantic models of trust. We also identified various models of trust that are computationally hard to satisfy. More specifically, we showed that several quantitative demands of interest in anonymous routing networks that require "at least k" nodes to be visited are NP-hard. Our model can be used to improve trustworthiness in organizational networks and deployed anonymizing networks such as Tor.

APPENDIX

A.1 Multiplicative combiner

LEMMA A.1. *Let s be the sum of weights on the a,b-path π in G'. The path confidence $\mathcal{C}_\pi$ of π in G is equal to e^{-s}.*

PROOF. *Let $c_1,\ldots,c_n$ be the confidence levels of all the nodes in π except a. $\mathcal{C}_\pi = c_1c_2\ldots c_n$ since $c_a = 1$. Now $s = \sum_{i=1}^{n} -\ln(c_i) = -\sum_{i=1}^{n}\ln(c_i) = -\ln(c_1c_2\ldots c_n)$. Hence $e^{-s} = e^{\ln(c_1c_2\ldots c_n)} = c_1c_2\ldots c_n = \mathcal{C}_\pi$.*

Note that if for there exists a $c_i = 0$, then the path confidence is 0. Moreover, $s = \infty$ since $-\ln(0) = \infty$ and $e^{-s} = 0$, so there is no discrepancy for confidence levels of 0. Essentially, any path which includes a node of 0 confidence will not be chosen by the user. □

LEMMA A.2. *For any two a,b-paths π_1,π_2 with total weights w_1,w_2 in G', we have $w_1 \le w_2$ if and only if $\mathcal{C}_{\pi_1} \ge \mathcal{C}_{\pi_2}$ in G.*

PROOF. *From Lemma A.1 we have that $w_1 \le w_2 \Leftrightarrow -w_1 \ge -w_2 \Leftrightarrow e^{-w_1} \ge e^{-w_2} \Leftrightarrow \mathcal{C}_{\pi_1} \ge \mathcal{C}_{\pi_2}$.* □

THEOREM A.3. *The k-shortest a,b-paths in G' correspond to the k a,b-paths of highest path confidence in G.*

PROOF. *This follows from Lemma A.2 since if we order all the a,b-paths in G' in increasing order of weight, they are ordered in decreasing order of path confidence in G.* □

A.2 Average and variance combiners

THEOREM A.4. *The Minimum Average Cost Simple-Path Problem (MinACSPP) is NP-hard.*

PROOF. *We reduce the s,t-HP to MinACSPP. Given a graph $G = (V,E)$, and vertices $s,t \in V$, assign the weight 1 to all vertices except t. Assign the weight $1+\delta$ to t. Any s,t-path of length (number of vertices) n will have average cost $\frac{(n-1)+(1+\delta)}{n} = 1+\frac{\delta}{n}$. This average cost is minimized for largest possible $n = |V|$. Hence the solution to MinACSPP will yield an s,t-path that visits $|V|$ vertices if and only if an s,t-Hamiltonian path exists in G. Hence MinACSPP is NP-hard.* □

THEOREM A.5. *The Maximum Average Cost Simple-Path Problem (MaxACSPP) is NP-hard.*

PROOF. *We reduce the s,t-HP to MinACSPP. Given a graph $G = (V,E)$, and vertices $s,t \in V$, assign the weight 1 to all vertices except t. Assign the weight $1-\delta$ to t (where $\delta < 1$). Any s,t-path of length (number of vertices) n will have average cost $\frac{(n-1)+(1-\delta)}{n} = 1-\frac{\delta}{n}$. This average cost is maximized for largest possible $n = |V|$. Hence the solution to MaxACSPP will yield an s,t-path that visits $|V|$ vertices if and only if an s,t-Hamiltonian path exists in G. Hence MaxACSPP is NP-hard.* □

THEOREM A.6. *The Minimum Variance Simple-Path Problem (MVSPP) is NP-hard.*

PROOF. *We reduce the s,t-HP to MVSPP. Given a graph $G=(V,E)$, and vertices $s,t \in G$, assign the weight 1 to all vertices other than t. Assign the weight $1+\delta$ to t.*

Any s,t-path of length (number of vertices) n will have variance

$$= \frac{(n-1)1^2+(1+\delta)^2}{n} - \frac{(n+\delta)^2}{n^2} \quad (1)$$

$$= \frac{n+\delta^2+2\delta}{n} - \frac{n^2+\delta^2+2n\delta}{n^2} \quad (2)$$

$$= \frac{n^2+n\delta^2+2n\delta}{n^2} - \frac{n^2+\delta^2+2n\delta}{n^2} \quad (3)$$

$$= \frac{\delta^2(n-1)}{n^2} \quad (4)$$

For any fixed δ, since $n \geq 3$ (by assumption that $(s,t) \notin E$), the variance is minimized for largest possible $n=|V|$. Hence the solution to MVSPP will yield an s,t-path that visits $|V|$ vertices if and only if an s,t-Hamiltonian path exists in G. Hence MVSPP is NP-hard. □

A.3 k-Distinct Vertex Minimum Weight Walk Problem (k-MWWP)

The Vertex Weighted k-Minimum Tree Problem is NP-hard [Fischetti et al. 1994]. We reduce this to s,t-k-VMT, and in turn to k-MWWP to prove NP-hardness of k-MWWP.

Definition A.7. **Vertex Weighted k-Minimum Tree Problem (k-VMT):** Given an undirected graph $G=(V,E)$, with positive vertex weights $w(v)$ for each vertex $v \in V$, find a tree T in G with $k<|V|$ vertices (we call this a k-tree[2]), where T is of minimum weight. The weight $w(T)$ of T is the sum of weights of the set of vertices in T. $w(T)=\sum_{v\in T} w(v)$.

Definition A.8. **Vertex Weighted s,t-k-Minimum Tree Problem (s,t-k-VMT):** Given an undirected graph $G=(V,E)$, with positive vertex weights $w(v)$ for each vertex $v \in V$, find a tree T in G with $k<|V|$ vertices containing specified vertices s and t. (we call this an s,t-k-tree), where T is of minimum weight. The weight $w(T)$ of T is the sum of weights of the set of vertices in T. $w(T)=\sum_{v\in T} w(v)$.

LEMMA A.9. *Vertex Weighted s,t-k-Minimum Tree Problem (s,t-k-VMT) is NP-hard.*

PROOF. *We reduce k-VMT to s,t-k-VMT.*

Given k and an undirected graph $G=(V,E)$ with vertex weights $w(v)$, construct $G'=(V',E')$ in the following way. Assume some ordering $v_1,\ldots,v_n$ of vertices in V. Start with a copy of G, and for each vertex $v_i \in V$ add two new vertices s_i and t_i. Add the edges (v_i,s_i) and (v_i,t_i). Let $M=\sum_1^n w(v_i)$. Assign the weight $M+1$ to s_i and t_i. In addition, add the vertices s and t and the edges (s,s_i) and (t,t_i) for all $i=1,\ldots,n$. Assign weights $\delta=1$ to s,t. This new graph contains $3n+2$

[2]Fischetti et al. [1994] define a k-tree to have k edges, however trees with k edges have $k+1$ vertices, and hence our definition is equivalent

vertices, and $|E| + 4|V|$ edges. G', along with s, t and $k+4$ is used as input to the s,t-k-VMT problem. We show that G has a k-tree of cost $\leq c$ if and only if G' has an s,t-$(k+4)$-tree of cost $\leq c + 2(M+1) + 2\delta$, where $c \leq M$.

Clearly if G has a k-tree T of cost $\leq c$, then we we can add vertices s_i, t_i for some $v_i \in T$, along with s, t and edges (v_i, s_i), (v_i, t_i), (s, s_i), (t, t_i) to obtain an s,t-$(k+4)$-tree T' in G', where $w(T') \leq c + 2(M+1) + 2\delta$.

Likewise, let T' be an s,t-$(k+4)$-tree in G' with weight $\leq c+2(M+1)+2\delta$, where $c \leq M$. We argue that T' contains k vertices from V, and hence includes only two vertices s_i and t_j for some particular values of i, j. Since $s, t \in V(T')$, we know that must be at least two such vertices. Consider the case when there are more than two such vertices. We have $w(T') \geq 3(M+1)+2\delta$. But since $w(T') \leq c+2(M+1)+2\delta$, and $c \leq M$, we have $w(T') < 3(M+1) + 2\delta$, which is a contradiction. Hence there are only two vertices of weight $M+1$. Let T be the k-vertex embedding of T' in G. We know that T is a tree because any two vertices in T are connected in T', but cannot be connected through s, t, s_i, s_j. T is therefore connected and is a tree. Furthermore $w(T) = w(T') - 2(M+1) - 2\delta \leq c$.

It follows that a minimum s,t-$(k+4)$-tree T' of G' can be transformed into the minimum k-tree of G, and we have that s,t-k-VMT is NP-hard. □

THEOREM A.10. *k-Distinct Vertex Minimum Weight Walk Problem (k-MWWP) is NP-hard.*

PROOF. *We reduce s,t-k-VMT, which is NP-hard from Lemma A.9 to k-MWWP.*

Given an undirected graph $G = (V, E)$ with vertex weights $w(v)$, k, and $s, t \in V$, create the directed graph $G' = (V' = V, E')$, where each undirected edge $(u, v) \in E$ is replaced by directed edges (u, v) and (v, u) in E'.

We claim that G contains an s,t-k-tree of weight $\leq c$ if and only if G' contains an s,t-k-walk of weight $\leq c$.

If T is an s,t-k-tree of weight $\leq c$. Consider the embedding T' of T in G', where each undirected edge (u, v) in T is replaced with the corresponding directed edges (u, v) and (v, u) in T'. T' is a strongly connected subgraph of G' with k distinct vertices. Hence we can construct a walk p from s to t using only vertices and edges in T', yielding an s,t-k-walk of the same weight, which is $\leq c$.

Let p be an s,t-k-walk of G' of cost $\leq c$. Consider the embedding G_p of p in G, where directed edges of p are replaced by undirected edges in G. G_p is a connected subgraph of G. Let T_p be a spanning tree of G_p (this can be computed in polynomial time). T_p is an s,t-k-tree of the same weight $\leq c$.

It follows that a minimum weight s,t-k-walk in G' can be transformed into a minimum weight s,t-k-tree in G, and hence k-MWWP is NP-hard. □

A.4 k^+-Distinct Vertex Minimum Weight Walk Problem (k^+-MWWP)

LEMMA A.11. *Minimum cost k^+-tree problem is NP-hard (k^+-VMT).*

PROOF. *We reduce k-VMT to k^+-VMT.*

Given a graph G we claim that there exists a k-tree T of cost $\leq c$ if and only if there exists a k^+-tree T' of cost $\leq c$.

Clearly, a k-tree T of cost $\leq c$ is also a k^+-tree of cost $\leq c$.

Now consider a k^+-tree T' of cost $\leq c$. Consider any subtree T of T' with k vertices. T is a k-tree with $w(T) \leq c$.

It follows that the minimum weight k^+-tree of G yields a minimum weight k-tree of G, and hence the k^+-tree problem is NP-hard. □

LEMMA A.12. *Vertex Weighted s,t-k-Minimum Tree Problem (s,t-k-VMT) is NP-hard.*

PROOF. *We reduce k^+-VMT to s,t-k-VMT.*

Given k and an undirected graph $G = (V, E)$ with vertex weights $w(v)$, construct $G' = (V', E')$ in the following way. Assume some ordering $v_1, \ldots, v_n$ of vertices in V. Start with a copy of G, and for each vertex $v_i \in V$ add two new vertices s_i and t_i. Add the edges (v_i, s_i) and (v_i, t_i). Let $M = \sum_1^n w(v_i)$. Assign the weight $M + 1$ to s_i and t_i. In addition, add the vertices s and t and the edges (s, s_i) and (t, t_i) for all $i = 1, \ldots, n$. Assign weights $\delta = 1$ to s, t. This new graph contains $3n + 2$ vertices, and $|E| + 4|V|$ edges. G', along with s, t and $k + 4$ is used as input to the s,t-k-VMT problem. We show that G has a k^+-tree of cost $\leq c$ if and only if G' has an s,t-$(k+4)^+$-tree of cost $\leq c + 2(M + 1) + 2\delta$, where $c \leq M$.

Clearly if G has a k^+-tree T of cost $\leq c$, then we we can add vertices s_i, t_i for some $v_i \in T$, along with s, t and edges (v_i, s_i), (v_i, t_i), (s, s_i), (t, t_i) to obtain an s,t-$(k+4)^+$-tree T' in G', where $w(T') \leq c + 2(M + 1) + 2\delta$.

Likewise, let T' be an s,t-$(k+4)^+$-tree in G' with weight $\leq c+2(M+1)+2\delta$, where $c \leq M$. We argue that T' contains at least k vertices from V, and includes only two vertices s_i and t_j for some particular values of i, j. Since $s, t \in V(T')$, we know that must be at least two such vertices. Consider the case when there are more than two such vertices. We have $w(T') \geq 3(M+1)+2\delta$. But since $w(T') \leq c+2(M+1)+2\delta$, and $c \leq M$, we have $w(T') < 3(M + 1) + 2\delta$, which is a contradiction. Hence there are only two vertices of weight $M + 1$. Let T be the embedding of T' in G using only the vertices in V. We know that T is a tree because any two vertices in T are connected in T', but cannot be connected through s, t, s_i, s_j. T is therefore connected and is a k^+-tree. Furthermore $w(T) = w(T') - 2(M + 1) - 2\delta \leq c$.

It follows that a minimum s,t-$(k+4)^+$-tree T' of G' can be transformed into the minimum k^+-tree of G, and we have that s,t-k^+-VMT is NP-hard. □

THEOREM A.13. *k^+-Distinct Vertex Minimum Weight Walk Problem (k^+-MWWP) is NP-hard.*

PROOF. *We reduce s,t-k^+-VMT, which is NP-hard from Lemma A.9 to k^+-MWWP.*

Given an undirected graph $G = (V, E)$ with vertex weights $w(v)$, k, and $s, t \in V$, create the directed graph $G' = (V' = V, E')$, where each undirected edge $(u, v) \in E$ is replaced by directed edges (u, v) and (v, u) in E'.

We claim that G contains an s,t-k^+-tree of weight $\leq c$ if and only if G' contains an s,t-k^+-walk of weight $\leq c$.

If T is an s,t-k^+-tree of weight $\leq c$. Consider the embedding T' of T in G', where each undirected edge (u, v) in T is replaced with the corresponding directed edges (u, v) and (v, u) in T'. T' is a strongly connected subgraph of G' with at least k distinct vertices. Hence we can construct a walk p from s to t using only vertices and edges in T', yielding an s,t-k^+-walk of the same weight, which is $\leq c$.

Let p be an s,t-k^+-walk of G' of cost $\leq c$. Consider the embedding G_p of p in G, where directed edges of p are replaced by undirected edges in G. G_p is a connected subgraph of G. Let T_p be a spanning tree of G_p (this can be computed in polynomial time). T_p is an s,t-k^+-tree of the same weight $\leq c$.

It follows that a minimum weight s,t-k^+-walk in G' can be transformed into a minimum weight s,t-k^+-tree in G, and hence k^+-MWWP is NP-hard. □

A.5 Scoped minimum average cost

THEOREM A.14. *k Minimum Average Cost Simple-Path Problem (k-MinACSPP) is NP-hard.*

PROOF. *We reduce s,t-HP to k-MinACSPP.*

Given a graph $G = (V, E)$, and vertices $s, t \in V$, simply specifying $k = |V|$, and $w(v) = 1$ for all $v \in V$, k-MinACSPP will return an s,t-Hamiltonian path in G if and only if it exists. □

THEOREM A.15. *k^+ Minimum Average Cost Simple-Path Problem (k^+-MinACSPP) is NP-hard.*

PROOF. *We reduce s,t-HP to k^+-MinACSPP.*

Given a graph $G = (V, E)$, and vertices $s, t \in V$, simply specifying $k = |V|$, and $w(v) = 1$ for all $v \in V$, k^+-MinACSPP will return an s,t-Hamiltonian path in G if and only if it exists. □

THEOREM A.16. *k^- Minimum Average Cost Simple-Path Problem (k^--MinACSPP) is NP-hard.*

PROOF. *We reduce s,t-HP to k^--MinACSPP.*

Given a graph $G = (V, E)$, and vertices $s, t \in V$, assign the weight 1 to all vertices other than t. Assign the weight $1 + \delta$ to t. Any s,t-path of length n will have average cost $\frac{(n-1)+(1+\delta)}{n} = 1 + \frac{\delta}{n}$. This average cost is minimized for largest possible $n = |V|$. Hence for $k = n$, the solution to k^--MinACSPP will yield an s,t-path that visits $|V|$ vertices if and only if an s,t-Hamiltonian path exists in G. Hence k^--MinACSPP is NP-hard. □

THEOREM A.17. *k Minimum Average Cost Walk Problem (k-MACWP) is NP-hard.*

PROOF. *We can reduce k-MWWP to k-MACWP.*

Given a graph $G = (V, E)$, vertices $s, t \in V$, and k, we need to find the minimum cost walk from s to t with exactly k vertices.

We use G, s,t and k as input to k-MACWP. Let $w(p)$ be the weight of a walk p, and $a(p)$ be the average cost of p. We have that $w(p) \leq c$ if and only if $a(p) \leq \frac{c}{k}$.

Hence the minimum average weight s,t-k-walk in G will be the minimum cost s,t-k-walk in G. □

Similarly the following problems can be shown to be NP-hard. We omit the proofs since they are similar to the proofs above.

Definition A.18. *k* **Maximum Average Cost Simple-Path Problem (k-MaxACSPP):** Given a graph $G = (V, E)$, with positive vertex weights $w(v)$ for

each vertex $v \in V$, and vertices $s, t \in V$, find an s,t-k-path p that maximizes the average cost of p. The **average cost** of a path p is defined as the the total additive cost of p divided by the number of vertices in p.

k^+ Maximum Average Cost Simple-Path Problem (k^+-MaxACSPP): Given a graph $G = (V, E)$, with positive vertex weights $w(v)$ for each vertex $v \in V$, and vertices $s, t \in V$, find an s,t-k^+-path p that maximizes the average cost of p. The **average cost** of a path p is defined as the the total additive cost of p divided by the number of vertices in p.

k^- Maximum Average Cost Simple-Path Problem (k^--MaxACSPP): Given a graph $G = (V, E)$, with positive vertex weights $w(v)$ for each vertex $v \in V$, and vertices $s, t \in V$, find an s,t-k^--path p that maximizes the average cost of p. The **average cost** of a path p is defined as the the total additive cost of p divided by the number of vertices in p.

k-Minimum Variance Simple-Path Problem (k-MVSPP): Given a graph $G = (V, E)$, with positive vertex weights $w(v)$ for each vertex $v \in V$, and vertices $s, t \in V$ such that $(s, t)\neg \in E$, find an s,t-k-path p that minimizes the variance of weights for the set of vertices in p.

k^+-Minimum Variance Simple-Path Problem (k^+-MVSPP): Given a graph $G = (V, E)$, with positive vertex weights $w(v)$ for each vertex $v \in V$, and vertices $s, t \in V$ such that $(s, t)\neg \in E$, find an s,t-k+-path p that minimizes the variance of weights for the set of vertices in p.

k^--Minimum Variance Simple-Path Problem (k^--MVSPP): Given a graph $G = (V, E)$, with positive vertex weights $w(v)$ for each vertex $v \in V$, and vertices $s, t \in V$ such that $(s, t)\neg \in E$, find an s,t-k^--path p that minimizes the variance of weights for the set of vertices in p.

ACKNOWLEDGMENT

We would like to thank Chandra Chekuri, John Fischer, Sariel Har-Peled, Viraj Kumar, Kevin Milans, Shripad Thite, Mahesh Viswanathan, and Erin Wolf for their help with some of the algorithms and proofs in this paper. We also thank William H. Sanders and Marianne Winslett for their helpful comments.

REFERENCES

SAPHIRE Project Homepage. `http://saphire.inel.gov/`.

Verisign Homepage. `http://www.verisign.com/`.

1981. Fault Tree Handbook, NUREG-0492. United States Nuclear Regulatory Commission.

Al-Muhtadi, J., Campbell, R. H., Kapadia, A., Mickunas, D., and Yi, S. 2002. Routing Through the Mist: Privacy Preserving Communication in Ubiquitous Computing Environments. In *Proceedings of The 22nd IEEE International Conference on Distributed Computing Systems (ICDCS)*. 74–83.

Alon, N., Yuster, R., and Zwick, U. 1995. Color-coding. *Journal of the ACM 42,* 4 (July), 844–856.

Andersen, D. G., Balakrishnan, H., Kaashoek, M. F., and Morris, R. 2001. Resilient Overlay Networks. In *Proc. 18th ACM SOSP, Banff, Canada*.

Bishop, M. 2003. Computer Security: Art and Science. Addison-Wesley, ISBN 0-201-44099-7.

Blum, C. and Ehrgott, M. 2003. Local search algorithms for the k-cardinality tree problem. *Discrete Appl. Math. 128,* 2-3, 511–540.

Camenisch, J. and Lysyanskaya, A. 2005. A Formal Treatment of Onion Routing. In *Proceedings of CRYPTO 2005*. Springer-Verlag, LNCS 3621, 169–187.

Chandra, R., Ramasubramanian, V., and Birman, K. P. 2001. Anonymous Gossip: Improving Multicast Reliability in Mobile Ad-Hoc Networks. In *Proceedings of the The 21st International Conference on Distributed Computing Systems*. IEEE Computer Society, Washington, DC, USA, 275.

Chaum, D. 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM 4,* 2 (Feb.).

Clarke, E., Grumberg, O., and Peled, D. 2000. *Model Checking*. MIT Press.

Coppit, D. and Sullivan, K. J. 2000. Galileo: A tool built from mass-market applications. In *Proceedings of the 22nd International Conference on Software Engineering, Limerick, Ireland*. 750–753.

Demri, S. and D'Souza, D. 2002. An Automata-Theoretic Approach to Constraint LTL. In *FST TCS '02: Proceedings of the 22nd Conference Kanpur on Foundations of Software Technology and Theoretical Computer Science*. Springer-Verlag, London, UK, 121–132.

Dingledine, R., Mathewson, N., and Syverson, P. 2004. Tor: The Second-Generation Onion Router. In *Usenix Security*.

Fischetti, M., Hamacher, H. W., Jörnsten, K., and Maffioli, F. 1994. Weighted k-Cardinality Trees: Complexity and Polyhedral Structure. *Networks 24*, 11–21.

Hershberger, J. E., Maxel, M., and Suri, S. 2003. Finding the k shortest simple paths: a new algorithm and its implementation. In *Proceedings, 5th Workshop Algorithm Engineering & Experiments (ALENEX)*. SIAM.

Jaffe, J. M. 1984. Algorithms for Finding Paths with Multiple Constraints. *Networks 14*, 95–116.

Jamoussi, B. et al. 2002. Constraint-Based LSP Setup using LDP. RFC 3212.

Kapadia, A., Naldurg, P., and Campbell, R. H. 2004. Routing with Confidence: Supporting Discretionary Routing Requirements in Policy Based Networks. In *Proceedings IEEE 5th International Workshop on Policies for Distributed Systems and Networks (POLICY 2004)*. 45–54.

Karger, D., Motwani, R., and Ramkumar, G. D. S. 1993. On Approximating the Longest Path in a Graph. In *Proceedings of WADS*. 421–432.

Mladenović, N. and Urošević, D. 2004. Variable neighborhood search for the k-cardinality tree. 481–500.

Qiu, L., Yang, Y. R., Zhang, Y., and Shenker, S. 2003. On Selfish Routing in Internet-Like Environments. In *Proc. of ACM SIGCOMM*.

Reed, M. G., Syverson, P. F., and Goldschlag, D. M. 1998. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communication (JSAC), Special Issue on Copyright and Privacy Protection 16,* 4, 482–494.

Reiter, M. K. and Rubin, A. D. 1998. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security*.

Sahner, R. A. and Trivedi, K. S. 1986. A Hierarchial, Combinatorial-Markov Model of Solving Complex Reliability Models. In *Proceedings of 1986 ACM Fall Joint Computer Conference*. IEEE Computer Society Press, Los Alamitos, CA, USA, 817–825.

Salsano, S. and Veltri, L. 2002. QoS Control by Means of COPS to Support SIP-Based Applications. *Special Issue on Policy-Based Networking 16,* 2 (Mar.).

Singh, A., Castro, M., Rowstron, A., and Druschel, P. 2004. Defending against Eclipse attacks on overlay networks. In *Proceedings of the 11th ACM SIGOPS European Workshop, Leuven, Belgium*.

Sloman, M. and Lupu, E. 2002. Security and Management Policy Specification. *Special Issue on Policy-Based Networking 16,* 2 (Mar.).

Tarjan, R. E. 1972. Depth first search and linear graph algorithms. *SIAM Journal on Computing 1,* 2, 146–160.

Wang, Z. and Crowcroft, J. 1996. Quality-of-Service Routing for Supporting Multimedia Applications. *IEEE Journal on Selected Areas in Communications 14*, 1228–1234.

Yen, J. Y. 1971. Finding the K shortest loopless paths in a network. In *Management Science*. Vol. 17. 712–716.

Yen, J. Y. 1972. Another algorithm for finding the K shortest loopless network paths. In *Proceedings of 41st Mtg. Operations Research Society of America*. Vol. 20.

Yi, S., Naldurg, P., and Kravets, R. 2001. Security-Aware Ad Hoc Routing for Wireless Networks. Poster presentation, ACM Symposium on Mobile Ad Hoc Networking & Computing (Mobihoc).

Yu, T., Winslett, M., and Seamons, K. E. 2003. Supporting Structured Credentials and Sensitive Policies through Interoperable Strategies in Automated Trust Negotiation. *ACM Transaction on Information and System Security*.

Zhang, L., Deering, S. E., Estrin, D., Shenker, S., and Zappala, D. 1993. RSVP: A New Resource ReSerVation Protocol. *IEEE Network 7,* 5 (Sept.), 8–18.

Zhou, L., Schneider, F. B., and van Renesse, R. 2002. COCA: A Secure Distributed On-line Certification Authority. *ACM Transactions on Computer Systems 20,* 4 (Nov.), 329–368.